

Open in app ↗

Sign up

Sign in

Medium

 Search Write

Blade Kotelly · Follow

8 min read · Oct 6, 2016



K-Scripts: The fastest and most flexible way to articulate a user experience

We always sketch and doodle whenever we make physical products. But can we *sketch* user experiences?

In designing physical products, sketching is commonly used as a quick and inexpensive method to allow the designer to express an idea quickly, evaluate the concept, and explain it to others. Early sketches (which really are a form of prototyping) can be easily tweaked, changed wholesale, edited, questioned or improved. Consider Da Vinci's helicopter concept, or Degas' studies of ballet dancers made before he created 3d sculptures — both are methods to explore and understand the idea. In the late 90s I realized that there was a need for UX designers to have a method to sketch — and I created a tool I call **K-Scripts**.

I started to apply this same idea to web design, mobile application design, social-robot interaction design, and teaching it in my design-thinking class at MIT. Since then, it has been used at numerous organizations including big ones like Adobe and the U.S. Air Force, for articulating interactive experiences for software, hardware, robotics and a whole lot more.

The power of sketching comes from the rapid expression of the idea as it's generated — which means that a designer doesn't have time to become emotionally attached on a particular strategy. We've all had experiences when someone has invested a lot of time creating an expression of an idea that the slightest question or request to change direction becomes a difficult, or sometimes antagonistic, conversation — while expressing the same questions or changes early in the process would have been met with a desire to work collaboratively.

When designers present overly-developed/overly-complete ideas, that are too close to the actual solution too early in the design process, people often reject the ideas or fail to understand the core ideas because it's easy to focus on the wrong details.

If you know about the Uncanny Valley, you'll know that the more something like a robot starts to resemble a human form, the more affinity people have for the robot *until* the robot gets *very* close to looking human — in which case we are repulsed by it. Why? Because the details are so close, the figure looks more like a corpse than a human, and humans are wired to see these slight differences and stay away (staying away from dead/diseased organisms is generally good for a species). Can you think about a time when you saw a designer create a high-fidelity mockup, (for a website? a concept for a video commercial? a 3D printed and painted physical object?) only to show a senior executive at a design review and have the exec focus on a trivial detail? “Is that how big the logo would be?” “That's not the right font.” “We never use that word in this context” — when in fact they should be focused on the bigger issues that the designer is working to validate. This is what I call the *Uncanny Valley of Design*.

So here's the question: how can we sketch complex systems that deal with user-interactions, some of which may not even have a clear physical representations, while expressing an idea in an intellectually and emotionally compelling way?

My solution is to use words instead of images to express the concept. When you read a book, you imagine a lot that isn't explicitly stated. It's why books provide an extremely rich visual and emotional impression in your mind, yet the movie-version always feels off when you've read the book, first. If we could use that same technique to enable a designer to create a very strong perspective that focused the reader on the right things, then the reader could:

- 1) Clearly imagine the vision of a future end state
- 2) Keep the reader focused on all the important things, and none of the nit-picks
- 3) Engaging the reader's imagination to provide feedback and input that improve the concept

I pioneered the technique in the late 90s when I was making speech-recognition systems. Creating, at the time, a 2-column script to express the core of the idea of a person using a speech-system felt like the right thing to do. It's easy for a customer to read the script, and it runs like a little play. Then, when I wanted to explain some concepts that wouldn't nicely fit on the script (like how a particular interaction was technically possible) I expanded the expression of the document to include another column that enabled me to add notes, and allowed others to ask clarifying questions, or provide ideas in that space.

How To Make A K-Script

K-Scripts are valuable when they focus on the 80% case — the most typical cases that users will encounter. They can also be used for failure conditions (in which a user may make mistakes, even if we don't expect to have it happen often) or any other context — but the best value is to show off a normal interaction and not a possible, but unlikely one.

The format of the K-Script can be elegant and visually designed, or as simple as a spreadsheet table. Regardless of the format used, all K-Scripts contain three essential elements:

Who	Observable Action	Unobservable Action / Notes
-----	-------------------	-----------------------------

Who:

The first section defines the actor, whomever is performing the action, and shows the sequential relationship between a user (or users) and the product. The *who* could be a generic person, a specific user, a product, a system or any entity that is involved in the interaction.

Observable Action:

This section describes an action that a user performs or an action that occurs by a system — as long as it's something a viewer could observe. Whether it is discussing a physical interaction (touching a screen, walking down stairs) or a verbal one (talking to a machine), a detailed and truthful

description matters here; the *What* section is what enables the reader to create a mental picture of the interaction.

Unobservable Action:

This is the section that enables a designer to help the reader understand why a situation may be occurring (motivation of the user, or notes about how the system or technology works to enable this interaction.) It's also a good place for others to pose questions about the interaction.

Example: A Basic K-Script

Let's take a look at a short K-Script for the design of a new ticketing kiosk at a large fair:

Who	Observable Action	Unobservable Action / Notes
Customer	Walks into the fair ground and sees a bunch of very modern touch-screen kiosks in a row, like at an airport. She sees the screen which has 2 buttons, "Buy Tickets", "Make dinner reservations"	In research, we found that people generally want these 2 options, and we expect to have a limited number of staff there to handle other kinds of requests
Customer	Clicks on "Buy tickets"	
Kiosk	The screen shows "Number of Adults?" with a "+" and "-" button and it's default is set to "1", and "Number of Children?" with a "+" and "-" button and it's default is set to "0" and a "Next" button	Prices aren't displayed now, because there's another screen which allows people to enter or scan discount codes/coupons
Customer	Hits the "Next" button	
Kiosk	Screen shows the price "\$24.00 Enter discount code, or scan a coupon" and a "Next" button	
Customer	Taps "Next"	
Kiosk	With the price still on the screen, directs the customer to swipe their credit card.	
Customer	Swipes the card	
Kiosk	Plays a success sound, shows a success message and a ticket is printed out	

K-Scripts are an exceptionally easy thing to construct — and it would take a lot longer to sketch it out visually. The K-Script format also allows someone else to edit and provide feedback:

The Revised K-Script With Markings by 2 Readers

Who	Observable Action	Unobservable Action / Notes
Customer	Walks into the fair ground and sees a bunch of very modern touch-screen kiosks in a row, like at an airport. She sees the screen which has 2 buttons, "Buy Tickets", "Make dinner reservations"	In research, we found that people generally want these 2 options, and we expect to have a limited number of staff there to handle other kinds of requests
Customer	Clicks on "Buy tickets"	Is this odd if they have a coupon for free tickets? How about just "Tickets" ?
Kiosk	The screen shows "Number of Adults?" with a "+" and "-" button and it's default is set to "1", and "Number of Children?" with a "+" and "-" button and it's default is set to "0" and a "Next" button	Prices aren't displayed now, because there's another screen which allows people to enter or scan discount codes/coupons what if the user made a mistake? Can they start over? why not show prices here as well? smart idea to have "1" as the default number of adults!
Customer	Hits the "Next" button	
Kiosk	Screen shows the price "\$24.00 Enter discount code, or scan a coupon" and a "Next" button	will people know how to scan a coupon? where's the scanner?
Customer	Taps "Next"	
Kiosk	With the price still on the screen, directs the customer to swipe their credit card.	are we going to be able to use ApplePay here?
Customer	Swipes the card	
Kiosk	Plays a success sound, shows a success message and a ticket is printed out	I love the idea of the sound — can we have more sounds in this interaction?
Customer	There's a "Done" button that takes them back to the home screen. A confirmation number is shown so that if tickets don't print out they can talk to someone about their transaction.	How about this? I just want to make sure we don't have issues if the printers don't work

People can ask detailed questions, make comments, or even *add* content.

The thoroughness of the *Unobservable Action / Notes* section is essential for a thorough K-Script that will be used to convey an interaction because it's the place where people can challenge particular assumptions or learn how a technology can drive a particular type of interaction.

When to use a K-Script

The ability to whip K-Scripts up quickly is what makes them a powerful tool early on in the design process. A K-Script is a great way to introduce a new idea, as it can provide a clear picture of the interaction to those who have no familiarity with the product, but it can also be useful throughout the entire design process. Just as they function as an easy way to convey basic functionality, they can constantly be refined and become more robust (and more complete) over time, while still maintaining their flexibility.

In addition to their use as a resource for the designer during product development, K-Scripts can also be used to demonstrate functionality to clients or investors. By combining a thorough and well-developed K-Script with a rudimentary prototype, customers can experience the feel of the system (through a functional prototype) as well as understand the depth and complexity at which the final system will operate. And since it's text and it's focused on the user experience, anyone with a little understanding of the context will be able to understand it — even non-designers.

Since K-Scripts are not meant to be all encompassing it's not necessary to create convoluted and overly complex scripts. The better use is to have the script focus on typical interactions and use more detailed logic diagrams to show non-typical interactions. However, additional K-Scripts should be

created for less common or fringe cases when the script format is the best way to express the idea. A set of K-scripts creates a holistic view of the system that can communicate to lots of different groups and provide vision that aligns teams.

Tips for successful K-Scripts

- Ensure that common scenarios are represented in the K-Scripts — the things that the user will most frequently encounter.
- The K-Scripts should represent the salient functionality of the product but do not have to exhaustively cover every single case; users should be able to accurately infer how the system will respond to simple variations.
- However, consider including Error Condition K-scripts to convey what happens if something goes wrong. What happens if a user isn't successful? What happens if they say something to a speech-system that's unintelligible? What if a users changes their mind and needs to make a change? These are situations that will certainly happen, so pick the common types of error conditions that help designers, technologists, etc. to understand how common errors are handled.

Useful Conventions

Over the years here are the conventions I've found the most useful when writing K-Scripts. Use quotation marks for buttons, speech-commands, and other specific actions that a user can take:

- For an iPhone app 3 buttons appear: "Video" "Audio" "Pictures"
- For a Speech Recognition System: You can say "Red", "Green" or "Blue".

Use **angle brackets** for notes for the user to read while reading the script, for **variables, non-textual elements and some actions**.

For and iPhone app:

- <Logo appears>
- “The weather in <Boston> is <67> degrees”
- “There <is/are> <1> <book/books> remaining.”

For Speech Systems

- <user presses zero>
- <NBC audio jingle plays>

More Sample K-Scripts

Let’s take a look at a short K-Script for Ethnographic Research for an interaction at a sandwich store:

Who	Observable Action	Unobservable Action / Notes
Customer	Walks up to the counter and sees a menu listing sandwiches. Looks at the deli guy and says, "I'll take a large meatball sub."	
Deli guy	Looks up and walks up to the customer "Ok, would you like cheese on that?"	
Customer	"...umm..."	This customer doesn't see a list of cheese options – which is common for many delis
Customer	"Yeah, do you have provolone?"	The customer guesses
Deli guy	"Yeah. Sauce on it too?"	
Customer	"Yeah."	The customer assumes it's tomato sauce based on context
Deli guy	"All right." The deli guy gets out a roll to begin making the sub and starts to scoop tomato sauce onto the sub	

This quick-to-make K-Script allows for a design to conduct ethnographic research and made a representation of an existing interaction. It's an easy starting place for another designer to review the interaction and suggest methods to improve the sub ordering experience.

K-Script Showing An Error Condition

Here the script shows how a speech-recognition system would be able to handle a situation in which the user doesn't reply to a question:

Who	Observable Action	Unobservable Action / Notes
System	Are you calling for technical support of a Whirlpool Product?	
User	Yes	
System	Just say the name of the product, for-	User barges in
User	It's an....	User drifts off
System	Sorry, I didn't understand that, say the name of the product you're calling about. For example "Washing Machine." Go ahead.	
User	User doesn't say anything
System	Ok, right now I'm asking for you to say the name of the product you need technical support for. If you don't know the name of the product or want to talk to someone, press zero and-	
User	<presses zero>	User barges in
System	Alright, Let me get you someone who can help. Hold on.	System transfers the call to a representative
Agent	Hi, it's Kim at Whirlpool, how...	

K-Scripts for Every Design Problem

K-Scripts should be easy to make and extremely powerful — I encourage you to try them out and see if it speeds things up and helps you communicate your ideas to others (and to yourself). Highlight parts you find valuable and if you've got questions just ask!

Learn more at www.bladekotelly.com

Design

UX

Speech Recognition

Design Thinking

Web Design



Written by Blade Kotelly

Follow

301 Followers

UX Design Strategist, Consultant, Educator www.bladekotelly.com

Recommended from Medium



Melody Koh 🧐 in Prototypr

The UX job market REALLY sucks right now

AMERZOH.COM SEATTLE, WA
Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by \$25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay



Alexander Nguyen in Level Up Coding

The resume that got a software engineer a \$300,000 job at Google.

Why you should pivot and change directions right NOW

1-page. Well-formatted.

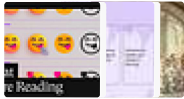
★ Aug 22 🖱 3.5K 💬 38



★ Jun 1 🖱 22K 💬 444



Lists



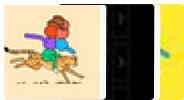
Stories to Help You Grow as a Designer

11 stories · 952 saves



Interesting Design Topics

257 stories · 800 saves



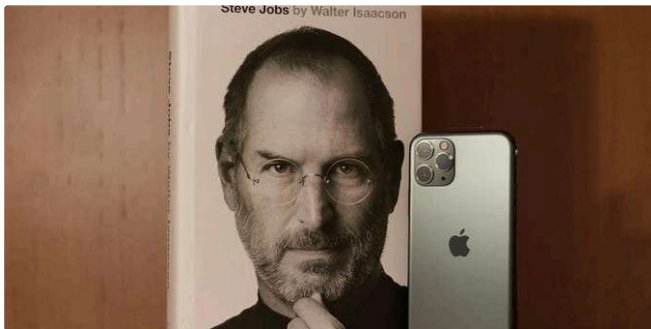
Figma 101

7 stories · 729 saves



Icon Design

36 stories · 419 saves



Dr. Derek Austin 🧠💻 in Career Programming

Steve Jobs Hated User Research. Here's Why I Agree With Him.

People have no idea what they want, and product managers would be better served b...

★ Jul 31 🖱 2.3K 💬 115



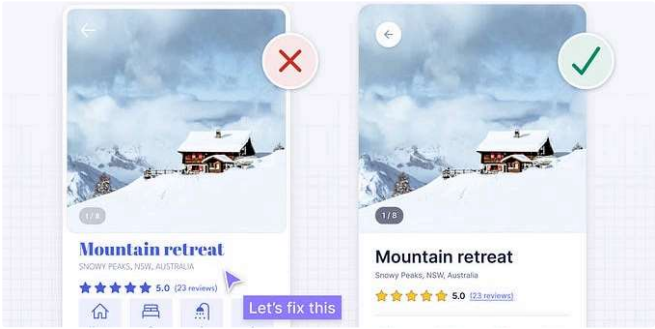
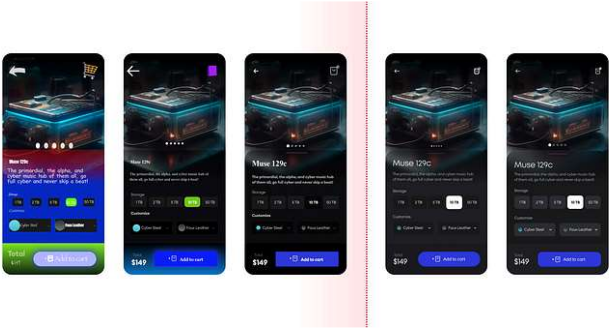
Michael F. Buckley in UX Collective



PayPal's generic brand refresh is a symptom of a troubling cultural...

The rise of minimalist corporate rebrands reflects a fundamental transformation in our...

★ 3d ago 🖱 1.3K 💬 38





 Michal Malewicz 

There are FIVE levels of UI skill.

Only level 4+ gets you hired.

 Apr 25, 2023  6.5K  72 

 Adham Dannaway  in UX Planet

16 little UI design tips that make a big impact

A step by step UI design case study to quickly fix an example user interface using logic...

Mar 14, 2023  31K  302 

See more recommendations