# Table of Contents

[Table of Contents](#_j5nwfjhver8g)

[Meta-Data](#_wmw6vju0tvzt)

[Lesson Goals](#_y0f332pzpsd3)

[Lesson Outcomes](#_yuun61fn00lj)

[Assessments](#_qy4klj35a395)

[Lesson Plan](#_jhf5z8xjau2m)

[Script](#_x1hgyjwhsibz)

[2.2.1 Introduction to Feedback Cycles](#_xgus6bu1y1vv)

[2.2.1.1 Headshot Studio](#_pg2fbhgqojjo)

[2.2.2 Feedback Cycles Are Fundamental](#_svyuskdb0fyr)

[2.2.2.1 Tablet Studio](#_rzvtj8z5kkkt)

[2.2.3 Gulf of Execution](#_rr89s7egnlr5)

[2.2.3.1 Tablet Studio](#_wrncu255shd)

[2.2.4 Gulf of Execution Example](#_jnx93y3chuxp)

[2.2.4.1 David’s House (Kitchen)](#_lfxnm8i5ittb)

[2.2.5 5 Tips: Gulfs of Execution](#_xza9oepw92vz)

[2.2.5.1 Headshot Studio](#_y86olqwahl8j)

[2.2.6 Gulf of Evaluation](#_715b7dsr77ml)

[2.2.6.1 Tablet Studio](#_kpkmy7k0x56q)

[2.2.7 Gulf of Evaluation Example](#_vwwlvqkz7udu)

[2.2.7.1 David’s House (Thermostat)](#_a63c8tbzo3bt)

[2.2.8 5 Tips: Gulfs of Evaluation](#_orthvxpfkimw)

[2.2.8.1 Headshot Studio](#_gcyqr1kems6f)

[2.2.8A Norman’s Feedback Cycle Stages](#_dfs1gjerc3ui)

[2.2.8A.1 Tablet Studio](#_ajae0m84zgaw)

[2.2.9 Good Design, Bad Design: Feedback Cycles](#_g3x8o9bm45dp)

[2.2.9.1 Headshot Studio (Morgan)](#_kwgbph200c0w)

[2.2.10 Reflections: Feedback Cycles](#_4yyjwadzjwt7)

[2.2.10.1 David’s House (Basement)](#_eyb9sfcdi1ye)

[2.2.10.2 Exercise](#_oc00vr4uhpv8)

[2.2.10.3 David’s House (Basement)](#_j2tpxmz6ir6t)

[2.2.11 Feedback Cycles in a Car 1](#_ogjm7pihmt1j)

[2.2.11.1 David’s House (Car)](#_gmswxtmjy0re)

[2.2.11.2 Exercise](#_gv94nqyg4ews)

[2.2.11.3 David’s House (Car)](#_crckhz1b095c)

[2.2.12 Feedback Cycles in David’s Car 2](#_kn6n1jjwyqh7)

[2.2.12.1 David’s House (Car)](#_4s1cmqjz2se1)

[2.2.12.2 Exercise](#_bgs4dahmrhe1)

[2.2.12.3 David’s House (Car)](#_bo68yvxe41rf)

[2.2.13 Feedback Cycles in a Car 3](#_5lrowqjbp4w)

[2.2.13.1 David’s House (Car)](#_7g7lmdxjhtsh)

[2.2.13.2 Exercise](#_l06ofkktdlpt)

[2.2.13.3 David’s House (Car)](#_xiy2kzozt7fa)

[2.2.14 Seven Questions for Bridging Gulfs](#_mszuntql2lm5)

[2.2.14.1 Headshot Studio](#_j3s4ue1ns9n5)

[2.2.15 Exploring HCI](#_9cs7uhinjqt7)

[2.2.15.1 Headshot Studio](#_vv2nyruzaxky)

[2.2.16 Design Challenge: Credit Card Readers 1](#_ynticnmbbpbs)

[2.2.16.1 David’s House (Breakfast Room)](#_cwlzzoej7gbf)

[2.2.16.2 Exercise](#_nh67xxkx2joa)

[2.2.16.3 David’s House (Breakfast Room)](#_od06zsvkvzsi)

[2.2.17 Design Challenge: Credit Card Readers 2](#_c6734k4ltxaf)

[2.2.17.1 David’s House (Breakfast Room)](#_c0fbldocpcd1)

[2.2.17.2 Exercise](#_ilgernxcpm3e)

[2.2.17.3 David’s House (Breakfast Room)](#_1cearh5ulxg1)

[2.2.18 Design Challenge: Credit Card Readers 3](#_mk5lygk87854)

[2.2.18.1 David’s House (Breakfast Room)](#_sfvhqr76vho)

[2.2.18.2 Exercise](#_qzlugojbrpau)

[2.2.18.3 David’s House (Breakfast Room)](#_9qffq999gk1n)

[2.2.19 Conclusion](#_bl6nc95h87ru)

[2.2.19.1 Headshot Studio](#_vwaijv7cpyeg)

# Meta-Data

## Lesson Goals

* Students will understand the overall nature of feedback in interaction design, including the feedback cycle between user and interface.
* Students will understand the definition of gulfs of execution.
* Students will understand the definition of gulfs of evaluation.

## Lesson Outcomes

* Students will be able to identify gulfs of execution and evaluation in task design.
* Students will be able to describe basic methods for bridging gulfs of execution and evaluation.
* Students will be able to identify subtle feedback cycles in their everyday lives.

## Assessments

* Students will complete a reflection exercise where they note the role of feedback cycles in their everyday lives.
* Students will reflect on the application of the lesson’s concepts to their chosen area of HCI.
* Students will engage in a short design task based on the lesson’s concepts.
* Students will complete a short answer assignment in which they (a) critique a provided interface, (b) select an interface to critique, and (c) design a revision of one of the critiqued interfaces.

## Lesson Plan

* The lesson begins by introducing students to the ubiquity of feedback cycles in everyday life and in interface design more narrowly.
* Then, the lesson describes the two gulfs to overcome in interface design: gulf of execution and gulf of evaluation.
* Each gulf is illustrated with a couple examples of good and bad gulfs of execution and evaluation.
* Students are then presented with questions to ask to evaluate how broad the gulf of execution and evaluation in a particular task.
* Students then reflect on the application of feedback cycles to the domain of HCI they chose at the beginning of the

# Script

## 2.2.1 Introduction to Feedback Cycles

### 2.2.1.1 Headshot Studio

* [C] David talking on left
* [A] Video clips of lesson playing on right
* [B] Lesson; Feedback Cycles
* Today we’ll be talking about a fundamental idea of HCI, **feedback** cycles.
* Feedback cycles are the way in which people interact with the world and then get feedback on the results of those interactions.
* [B] Topic; Ubiquity of feedback cycles
* [B] Definition; Ubiquity: <definition>
* We’ll talk about the **ubiquity** of feedback cycles.
* [B] Topic/Definition; Gulf of execution: the distance between a user’s goals and the actions required to realize those goals
* Then, we’ll talk about the gulf of **execution**, which is the distance between a user’s goals and the execution of the actions required to realize those goals.
* [B] Topic/Definition; Gulf of evaluation: the distance between the effects of those actions and the user’s understanding of the results
* Then, we’ll talk about the gulf of **evaluation**, which is the distance between the effects of those actions and the user’s understanding of the results.
* [B] Topic: Seven questions for designing feedback cycles
* We’ll discuss the seven **questions** we should ask ourselves when designing feedback cycles for users.
* [B] Topic: Everyday applications of feedback cycles
* We’ll also look at **applications** of these in multiple areas of our everyday lives.

## 2.2.2 Feedback Cycles Are Fundamental

### 2.2.2.1 Tablet Studio

* [V] The feedback cycle diagram
* Feedback cycles are incredibly ubiquitous
* [V] Fade out interface and computer in the diagram, replacing the computer with a textbook
* They’re how we learn everything from how to walk to how to solve a Rubik’s cube to how to take the third order partial derivative of a function. (I assume, I’ve never done that last one.)
* [V] Replace textbook one by one with car, computer, other person in order.
* Feedback cycles are embedded in how we drive, in how we cook, in how we interact with other people.
* We do something, we see the result, and we adjust what we do next time accordingly.
* You may have seen some other examples before, too.
* [V] Feedback cycle between a Cognitive System and The World from KBAI
* If you’ve taken Ashok’s and my Knowledge-Based AI class, we talk about how agents are constantly interacting with, learning from, and affecting the world around them.
* [V] Feedback cycle from Cyber-Physical Systems
* If you’ve taken Raheem Beyah’s Cyber-Physical Systems course, you’ve seen this without a human involved at all, as a system autonomously reads input and reacts accordingly.
* Under a definition, some would call this artificial intelligence, specifically because it mimics what a human does: act in the world and interpret the outcome.
* [V] [Some definitions of intelligence appear](http://www.vetta.org/documents/A-Collection-of-Definitions-of-Intelligence.pdf).
* In fact, if you look at some definitions of intelligence, you’ll find many either define feedback cycles or define an ability gained through engagement in a feedback cycle.
* [V] Feedback cycle diagram
* We’ll find that nearly all of HCI can be interpreted in some ways as an application of feedback cycles, whether between a person and a task, a person and an interface, or systems comprised of people and interfaces.

## 2.2.3 Gulf of Execution

### 2.2.3.1 Tablet Studio

* [V] Feedback cycle diagram
* In our feedback cycle diagram, we have on the left some user and on the right some task or system.
* The user puts some input into the system through the interface, and the system communicates some output back to the user via the interface.
* Incumbent on this are two general challenges.
* [V] Label appears for ‘Gulf of Execution’ by the input
* The first is the gulf of execution.
* The gulf of execution can be defined as: how do I know what I can do?
* The user has some goals.
* How do they figure out what to *do* to make those goals a reality?
* This is the gulf of execution: how hard is it to do in the interface what is necessary to accomplish those goals?
* Or, alternatively: what’s the difference between what the user thinks they should have to do, and what they actually have to do?
* There are a number of components of this.
* [V] First component, “Identify intentions” appears inside/instead of/etc. the arrow for gulf of execution -- possibly something like [this](https://interactioncultureclass.files.wordpress.com/2012/01/hutchinhollannorman.png)?
* They need to be able to identify what their goal is in the context of the system.
* There might be a mismatch between their own understanding and the system’s structure.
* Think of transitioning from an old-fashioned VCR to a more modern DVR, or from a DVR to watching things on demand.
* The user needs to think of their goal in terms of their current system.
* [V] Second component, “Identify actions”
* Second, they need to be able to identify the actions necessary to accomplish their goals.
* [V] Third component, “Execute in interface”
* And third, they need to be able to actually interface with the system to carry out the actions.

## 2.2.4 Gulf of Execution Example

### 2.2.4.1 David’s House (Kitchen)

* [C] David by the microwave
* Let’s take a simple example of gulf of execution.
* I’m making lunch. I have my bowl of chili.
* My goal is simple. I want to heat it up.
* How hard is it?
* Well, typically when I’ve been cooking in the past, cooking is defined in terms of the amount of time it takes.
* So, in the context of the system, I specify my intent as to microwave it for one minute.
* Now, what are the actions necessary to do so?
* <David does so>
* I press this button to open the door.
* I insert the dish and close the door.
* I press Time Cook to enter the ‘timed cooking’ mode.
* I enter the time, one minute.
* And I press start.
* I specified my intent: microwave for one minute.
* I specified my actions: pressing the right sequence of buttons.
* And I executed those actions: I pressed them.
* Could we make this better?
* There were a lot of button presses to microwave for one minute.
* If we think that’s a common behavior, we might be able to make it simpler.
* Instead of pressing Time Cook, 1, 0, 0, and Start… I just press 1 and wait.
* <David does so>
* So I’ve narrowed the gulf of execution by shrinking the number of actions required.
* However, I may have enlarged it by making it more difficult to identify the actions required.
* When I look at this, Time Cook gives me an idea of what that button does.
* So, if I’m a novice at this, I can discover how to accomplish my goal. That’s good for the gulf of execution: it’s easier to look at the button and figure out what to do than have to go find the manual and read about it.
* But once you know that all you have to do is press 1, that’s much easier.
* That’s something nice about this interface: it caters to both novices and experts. There’s a hard, discoverable way and a short, “invisible” way.
* But let’s rewind all the way back to what I said initially.
* My goal was to heat up my chili.
* I specified my intent in terms of this system as microwaving it for one minute.
* But was that the right thing to do?
* After one minute, my chili might not be hot enough.
* This microwave has an automatic reheat function that senses the food’s temperature and stops when the time seems right.
* So, the best bridge over the gulf of execution might also involve helping me reframe my intention.
* Instead of wanting to microwave for one minute, it might encourage me to reframe this as simply heating until ready, and letting the microwave do the rest.

## 2.2.5 5 Tips: Gulfs of Execution

### 2.2.5.1 Headshot Studio

* [C] David talking
* Here are five quick tips for bridging gulfs of execution.
* [B] 1. Make functions discoverable.
* **1. Make functions discoverable.** Imagine a user is sitting in front of your interface for the very first time. How would they know what they can do? Do they have to read the documentation? Take a class? Ideally, the functions of the interface would be discoverable, meaning that they can find them, clearly labeled, within the interface.
* [B] 2. Let the user mess around.
* **2. Let the user mess around.** You want your user to poke around and discover things, so make them feel safe doing so. Don’t include any actions that can’t be undone. Avoid any buttons that can irreversibly ruin their document or setup. That way, the user will feel safe discovering things in your interface.
* [B] 3. Be consistent with other tools.
* **3. Be consistent with other tools.** We all want to try new things and innovate, but we can bridge gulfs of execution nicely by adopting the same standards many other tools use. Use Ctrl+C for Copy, Ctrl+V for paste. Using a diskette icon for ‘save’, even though no one has used floppy disks in years. This makes it easy for users to figure out what to do in your interface.
* [B] 4. Know your user.
* **4. Know your user.** The gulf of execution has a number of components: identifying your intentions, identifying the actions to take, and taking the actions. For novice users, identifying their intentions and actions are most valuable, so making commands discoverable through things like menus is preferable. For experts, though, actually *doing* the action is more valuable. That’s why many experts prefer the command-line: although it lacks many usability principles targeted at novices, it’s very efficient.
* [B] 5. Feedforward.
* **5. Feedforward.** We’ve talked about feedback, which is a response to something the user did. Feedforward is more like feedback on what the user might want to do. It helps the user predict what the result of an action will be. For example, when you pull down on the Facebook newsfeed on your phone, it starts to show the refresh icon -- if you don’t finish pulling down, it doesn’t refresh. That’s feedforward: information on what will happen if you keep doing what you’re doing.
* [B] Lesson: Design Principles and Heuristics
* Many of these tips are derived from some of the fundamental principles of design pioneered by people like Don Norman and Jakob Nielsen, and we **cover** them more in another lesson.

## 2.2.6 Gulf of Evaluation

### 2.2.6.1 Tablet Studio

* [V] Label appears for ‘Gulf of Evaluation’ by the output.
* At the bottom of the feedback cycle, there is a gulf of evaluation.
* The gulf of evaluation can be described as, ‘How do I know what happened?’
* The user has taken some action.
* Now, they need to evaluate whether or not their action led to the desired goals.
* Whether or not they succeeded is actually inside the system.
* This isn’t about whether the action *did* succeed, but how the user becomes aware *that* their action succeeded.
* There are a few components to this as well.
* [V] First component, “Interface output”, appears similar to that in the gulf of execution video.
* First, there’s what’s actually displayed or communicated, the actual output.
* That might be something visual, a sound, a vibration, some kind of output.
* [V] Second component, “interpretation”, appears.
* Given that output, there is then some interpretation phase.
* The user needs to interpret the output to find out what it means for the system.
* [V] Third component, “evaluation”, appears.
* Then, once they have received and interpreted the output, they can evaluate whether the desired change occurred.

## 2.2.7 Gulf of Evaluation Example

### 2.2.7.1 David’s House (Thermostat)

* [C] David by thermostat
* So, let’s take a thermostat for example. I have a goal of making the room warmer.
* So, I do something in the thermostat to make the room warmer.
* What does the system do as a result? It turns the heat on. That would be the successful result of that action.
* But how do I *know* that the heat has turned on?
* Well, maybe I can hear it -- but that’s a one-time thing and it might be kind of quiet. If I miss hearing it, then I have no way of double-checking.
* So if I’m not sure if I heard it, I have to go and find a vent and put my hand on it and try to feel if the heat is coming out.
* And there’s more going on in a heater. It might have worked, but the heater doesn’t turn on immediately for one reason or the other.
* These are signs of a large gulf of evaluation.
* Neither the sound nor the vent are optimal displays because they’re either hard to reach or possible to miss.
* Feeling the heat might be easy to interpret, but hearing the heater turn on might not.
* So, either way, I have to do a lot to evaluate whether or not my actions were successful.
* And this is all for a very small piece of feedback. Ideally, if I wasn’t successful, we want the system to also tell me *why*, so I can evaluate what I did wrong and respond accordingly.
* There is a very large gulf of evaluation.
* So, how can we resolve it?
* [A] ‘Heat On’ appears on thermostat.
* Simple! We just mark that the heat is on.
* That sounds trivial, but nothing in the fundamental design of the system demanded a note like this.
* It’s only in thinking about how the system interfaces with a person that we find that need.
* I can let you know as well, this system still isn’t ideal.
* For various reasons, it will turn off the heater or air conditioning even when it hasn’t reached the temperature I put in.
* And it gives me no indication of why.
* I can look at the system and evaluate that the temperature is set to a lower temperature than that in the room.
* But at the same time, I can see that the heater isn’t on.
* Under those circumstances, I have no way of knowing if the heater is malfunctioning, if it has to switch off to recover, or…
* [A] David switches modes
* If it’s simply set to the wrong mode. The mode is visible, but I have to remember to check that if it appears to be malfunctioning.
* We can imagine an alternative message on the screen indicating the direction of the relationship or something similar that would give some sign that it’s set incorrectly.

## 2.2.8 5 Tips: Gulfs of Evaluation

### 2.2.8.1 Headshot Studio

* [C] David talking
* Here are five quick tips for bridging gulfs of evaluation.
* [B] 1. Give feedback constantly.
* **1. Give feedback constantly.** Don’t automatically wait for whatever the user did to be processed in the system before giving feedback. Give them feedback that input was received. Give them feedback on *what* input was received. Help the user understand where the system is in executing their action by giving feedback at every step of the process.
* [B] 2. Give feedback immediately.
* **2. Give feedback immediately.** Let the user know they have been heard even when you’re not ready to give them a full response. If they tap an icon to open an app, there should be immediate feedback on that tap. That way, even if the app takes a while to open, the user knows that the phone recognized their input. That’s why icons briefly grey out when you tap them on your phone.
* [B] 3. Match the feedback to the action.
* **3. Match the feedback to the action.** It seems like this amount of constant, immediate feedback would get annoying, and it would if executed poorly. Subtle actions should have subtle feedback, significant actions should have significant feedback.
* [B] 4. Vary your feedback.
* **4. Vary your feedback.** It’s often tempting to view our designs as existing solely on the screen, and so we want to give feedback on the screen. But the screen is where the interaction is taking place, so visual feedback can get in the way. Think about how auditory or haptic feedback can be used instead of relying just on visual feedback.
* [B] 5. Leverage direct manipulation.
* **5. Leverage direct manipulation.** We’ll talk about this a lot more, but whenever possible, let the user feel like they’re directly manipulating things in the system. Things like dragging stuff around or pulling something larger or smaller are very intuitive actions because they feel like you’re interacting directly with the content. Use that.
* [B] Lesson: Design Principles and Heuristics
* Again, we talk far more about these principles **elsewhere** in this unit, but they’re worth mentioning here as well. By loading them into your short-term memory several times, we help solidify them in your long-term memory… which, conveniently, is something else we talk about in this unit as well.

## 2.2.8A Norman’s Feedback Cycle Stages

### 2.2.8A.1 Tablet Studio

* [V] Norman’s diagram
* In Don Norman’s Design of Everyday Things, he provides a different way of looking at the same information.
* This diagram puts a greater emphasis on what the *user* is doing at each of these stages.
* The user is planning, specifying, performing, perceiving, interpreting, and comparing.
* This diagram also changes our terminology a bit, with good reason: we see bridges of execution and evaluation because these behaviors are specifically ways to *bridge* the gap between the goal in the user and the state of the world.
* Norman uses these stages to introduce seven questions we need to ask when designing our interfaces.
* [V] Seven questions appearing one-by-one
* 1. **How** easily can one determine the function of the device? This relates to the user’s goal: how easily can the user determine that the interface is capable of accomplishing their goal?
* 2. **How** easily can one tell what actions are possible? This is important for the user to be able to construct their plan.
* 3. **How** easily can one determine the mapping from intention to physical movement?
* 4. **How** easily can one actually perform that physical movement?
* 5. **How** easily can one tell what state the system is in?
* 6. **How** easily can one tell if the system is in the desired state?
* 7. **How** easily can one determine the mapping from system state to interpretation?
* You’ll notice these match our stages from earlier.
* Or, to make it even clearer, he also describes these stages with simple questions the user would be asking themselves.
* [V] Other questions appear
* (read through them)
* [V] Reflective/behavioral/visceral appear
* Norman also further articulates this by breaking the process into phases that span both execution and evaluation.
* The raw action and perception is referred to as visceral: this is the physical act of performing the plan, or the perceiving the outcome.
* The behavioral area is where we think about what steps to actually take or what we’re seeing from the interface.
* The reflective area is where we put it in the context of our goal: either translating a goal into a plan, or comparing the interpreted results to the original goal.
* [V] KBAI appears
* If you’ve taken CS7637: Knowledge-Based AI, this probably feels a little familiar. If you haven’t, I highly recommend it… but I might be a little bit biased.

## 2.2.9 Good Design, Bad Design: Feedback Cycles

### 2.2.9.1 Headshot Studio (Morgan)

* [C] Morgan tapping on her phone
* Good design: a phone that quietly ‘taps’ every time a letter is successfully pressed to let you know the press has been received.
* Bad design: a phone that loudly shouts the letter you just typed.
* Remember, small actions get small feedback. The only way you might want your device to yell a confirmation at you is if you just ordered a nuclear launch or something.

## 2.2.10 Reflections: Feedback Cycles

### 2.2.10.1 David’s House (Basement)

* [C] In David’s basement
* Let’s pause for a second and reflect on the roles of gulfs of execution and gulfs of evaluation in our own lives.
* Try to think of a time when you’ve encountered a wide gulf of execution and a wide gulf of evaluation.
* This doesn’t have to be a computer, it could be any interface.
* In other words, what was a time when you were interacting with an interface, but couldn’t think of how to accomplish what you wanted to accomplish?
* What was a time when you were interacting with an interface, and couldn’t tell if you’d accomplished what you wanted to accomplish?

### 2.2.10.2 Exercise

* [A] Two questions above with answer boxes

### 2.2.10.3 David’s House (Basement)

* [C] In David’s basement
* So, it’s not a coincidence that I’m filming this in my basement.
* This happened to me a few weeks ago. The circuit to our basement was tripped, which is where we keep our modem.
* So, our internet was out.
* This is a new house, and it was the first time we’d had a tripped breaker.
* So, pulled out my flashlight and opened the panel.
* None of the labels over here clearly corresponded to the breakers over here.
* I ended up trying every single one of them.
* Still, it didn’t work.
* Why?
* <Points to the tiny button by the floor>
* In reality, that was the reset button that had to be pressed.
* Terribly large gulf of execution.
* I knew what I wanted to accomplish, and I could translate it into this system’s terms easily.
* But figuring out the actions to accomplish that goal was very difficult.
* <David hesitates, turns to Amanda>
* Was that good?
* [Amanda] Oh, I don’t know, I wasn’t really watching.
* ...you weren’t watching?
* So I have no way of knowing if that was any good or not?
* Well isn’t that a… terrible gulf of evaluation.
* I joke, but lack of feedback on your performance at a task, whether it be filming like I’m doing now or doing a project like you’ll do later in our material, presents the same kind of poor gulf of evaluation.

## 2.2.11 Feedback Cycles in a Car 1

### 2.2.11.1 David’s House (Car)

* Fifteen years ago, we might not have been talking about cars when learning HCI.
* Now, however, many new cars are effectively computers on wheels.
* So, let’s look at how a feedback cycle applies to starting up a car.
* So, I’m in my car. My car uses a push button start.
* <David points to button>
* First question: why is the button located right here?

### 2.2.11.2 Exercise

* [V] Question: Why is the start button located here?
* [V] A. It’s closest to the engine, meaning the car will start faster.
* [V] B. It’s the only open place on the car after the other devices are inserted.
* [V] C. It’s where the user expects the button to be.
* [V] D. This location avoids the car getting accidentally turned off or on.

### 2.2.11.3 David’s House (Car)

* [C] David at his car
* Before cars had push-button start, this was where the key was inserted into the ignition.
* Why? I have no idea.
* But I do know that now, there are any number of places the start button could be placed.
* So why do we put it in the same place where we’ve always put it?
* Because it’s where the driver expects it to be.
* We help them across the gulf of execution by designing the system the way the user expects it to be designed.
* It makes it easier to translate intentions into actions.
* Now, other times we might violate this principle because of some other benefits, but when all else is equal, designing a system the way a user expects it to be designed is helps them across that gulf of execution.

## 2.2.12 Feedback Cycles in David’s Car 2

### 2.2.12.1 David’s House (Car)

* [C] David at his car
* So, we know where the ignition button is.
* Let’s press it.
* <David presses the ignition: a sound will play>
* Do you think the car turned on?

### 2.2.12.2 Exercise

* [V] Question: Do you think the car turned on?
* [V] A. Yes
* [V] B. No

### 2.2.12.3 David’s House (Car)

* [C] David at his car
* Well, what do we know.
* We know the car was off.
* We know this is clearly the on button based on where it’s located and how it’s labeled.
* And, most importantly, when we pressed it we heard kind of a happy sound.
* Did the car turn on?
* ...actually, no, it didn’t.
* To turn the car on, you have to press the brake pedal while pressing the on button.
* The car doesn’t do a great job of helping us across that gulf of execution: there’s no indicator that you’ve done it wrong until you’ve done it wrong.
* However, the car does give us a short gulf of evaluation.
* [S] Cut to the dashboard
* If you press the ‘on’ button without pressing the brake pedal, you’ll see an alert on the dashboard telling you what to do.
* The output presented is easy to interpret, and the context in which it is given helps us evaluate pretty quickly.
* So we have some trouble here with the gulf of execution, but the gulf of evaluation is still pretty short.
* So, now that I see this message, I press down the brake, press the button, and…
* <David presses down the brake, presses the button, and embarrassing music starts blaring>

## 2.2.13 Feedback Cycles in a Car 3

### 2.2.13.1 David’s House (Car)

* [C] David at his car
* So, now that we’ve seen the way the feedback cycle currently works, let’s talk about improving it.
* How might we make this feedback cycle better? How might we narrow the gulf of execution and gulf of evaluation?

### 2.2.13.2 Exercise

* [V] What are some ways this feedback cycle could be improved?
* [V] (answer box for student answers)

### 2.2.13.3 David’s House (Car)

* [C] David at his car
* So, here are a few ideas I had.
* We know that the screen can show an alert that the brake needs to be depressed to turn the car on. Why not show that immediately after the car door opens when the car is off?
* That narrows the gulf of execution for even a first-time driver, without lengthening it for an expert.
* Something that really throws me off, though, is that the car makes the same happy sound whether it turns on successfully or not.
* Watch.
* <David turns the car on without pressing the brake>
* That didn’t work.
* <David turns the car on with pressing the brake.>
* That worked.
* Either way, it plays the same initial sound, then it plays a different sound depending on whether it started up all the way.
* I understand the rationale behind this: the initial sound is a confirmation that the button was pressed, while the second sound confirms the car turned on.
* Personally, though, I think it would be better to simply use *any* sound to confirm the button was pressed, and differ the sound based on whether the car turned on.

## ~~2.2.14 Seven Questions for Bridging Gulfs~~

### ~~2.2.14.1 Headshot Studio~~

* ~~[A] Items numbered one through seven~~
* ~~In his book Design of Everyday Things, Don Norman outlines seven questions that we should ask when determining how usable a device is.~~
* ~~[B] Question text~~
* ~~1.~~ **~~How~~** ~~easily can one determine the function of the device?~~
* ~~[B] Question text~~
* ~~2.~~ **~~How~~** ~~easily can one tell what actions are possible?~~
* ~~[B] Question text~~
* ~~3.~~ **~~How~~** ~~easily can one determine the mapping from intention to physical movement?~~
* ~~[B] Question text~~
* ~~4.~~ **~~How~~** ~~easily can one actually perform that physical movement?~~
* ~~[B] Question text~~
* ~~5.~~ **~~How~~** ~~easily can one tell what state the system is in?~~
* ~~[B] Question text~~
* ~~6.~~ **~~How~~** ~~easily can one tell if the system is in the desired state?~~
* ~~[B] Question text~~
* ~~7.~~ **~~How~~** ~~easily can one determine the mapping from system state to interpretation?~~
* ~~You’ll notice these match our stages from earlier.~~
* ~~[B] Interpret~~
* **~~Interpret~~** ~~your goal in the context of the device’s function.~~
* ~~[B] Discern~~
* **~~Discern~~** ~~what actions are possible in the device.~~
* ~~[B] Identify~~
* **~~Identify~~** ~~how to perform an action or what action to perform.~~
* ~~[B] Perform~~
* **~~Perform~~** ~~the action.~~
* ~~[B] Observe~~
* **~~Observe~~** ~~the system’s output.~~
* ~~[B] Compare~~
* **~~Compare~~** ~~the output to the desired state.~~
* ~~[B] Interpret~~
* **~~Interpret~~** ~~the difference between the output and the desired state.~~

## 2.2.15 Exploring HCI

### 2.2.15.1 Headshot Studio

* [C] David talking
* I asked you earlier to pick an area of HCI in which you’re interested, and reflect on it throughout this course.
* Depending on the area you selected, feedback cycles can play a huge number of different roles.
* In healthcare, for example, feedback cycles are critical to helping patients manage their symptoms, and that relies on the results of certain tests being easy to interpret and evaluate.
* Feedback cycles also present on of the bigger challenges for gesture-based interactions.
* It can be difficult to get feedback on how the system interpreted a certain gesture and why it interpreted it that way.
* Compare that to touch where it’s generally very easy to understand where you touched a screen.
* So, think for a moment about how feedback cycles affect that area you chose to emphasize.

## 2.2.16 Design Challenge: Credit Card Readers 1

### 2.2.16.1 David’s House (Breakfast Room)

* [C] David talking at the counter
* Lately, I’ve encountered another interesting example of feedback cycles in action.
* You may have as well, they’re new credit card readers.
* My wife sells arts and crafts at local events, and so she has these Square readers that can scan credit cards.
* One version lets you swipe, the other lets you insert.
* So check this out.
* <David swipes his card>
* Traditionally, card readers work by swiping.
* The problem is that there’s typically no feedback on whether you’re swiping correctly.
* And what’s more, you can be wrong in both directions: you can be both too fast or too slow.
* So, you may have had a time when you were trying to swipe a card and kept doing so more and more slowly and deliberately, thinking that you’d done it too fast before.
* Then you discover, no, you’ve been going too slowly all along!
* There’s no feedback, and the space of acceptable input is bounded on both sides.
* You have to go above one speed and below another speed.
* But now, readers are moving to a model where you insert the card rather than slide them.
* In terms of feedback cycles, what are some ways that this system is better?

### 2.2.16.2 Exercise

* [V] What are some ways inserting the card gives a better feedback cycle than sliding it?
* [V] (box for student answer)

### 2.2.16.3 David’s House (Breakfast Room)

* First, in terms of the gulf of execution, the insertion method is easier to do.
* While you can be both too fast and too slow with the sliding method, you can’t be too far in with the insertion method.
* So you know if there’s an error, it’s because the card isn’t far enough in.
* Second, there’s rich feedback with the insertion method.
* It doesn’t even have to come from the screen.
* You feel the card stop when it’s far enough in.
* You have immediate physical feedback on whether you’re doing it correctly, rather than delayed feedback asking you to try again.

## 2.2.17 Design Challenge: Credit Card Readers 2

### 2.2.17.1 David’s House (Breakfast Room)

* So, using the insertion method is significantly easier.
* <David walks away>
* <David walks back>
* ...however, the insertion method introduces a new problem.
* With the sliding method, I never had to let go of my card, so there was little chance of me walking away without it.
* With the insertion method, I insert and wait.
* I’m not used to having to remember to retrieve my card.
* This isn’t quite as big a deal with these portable readers, but for the mounted ones, it can be far more problematic.
* So, how can we build some feedback into this system to make sure people remember their cards?

### 2.2.17.2 Exercise

* [E] How can we build feedback into this system to prevent customers from walking away without their credit cards?
* [E] (box for student answer)

### 2.2.17.3 David’s House (Breakfast Room)

* There are a few things we could do here.
* You might build in some kind of buzzer when the customer can take their card out to remind them to take it.
* ATM machines often do this, they’ll ring a buzzer until the card and cash are removed.
* But that’s noisy and potentially irritating.
* We could do something super complicated, like pair the credit card with a smartphone and ring the phone when it gets too far from the credit card.
* But that involves adding technology to every single credit card, a huge expense.
* So what about something simpler?
* Why not force the customer to remove their credit card to get their receipt and goods? Unless they’re going to walk away without what they bought, that will help ensure they remember their card.

## 2.2.18 Design Challenge: Credit Card Readers 3

### 2.2.18.1 David’s House (Breakfast Room)

* Now, notice one last thing about this example.
* We’ve been discussing how to make the process of sliding or swiping a credit card easier.
* What’s wrong with that question?

### 2.2.18.2 Exercise

* [E] What’s wrong with asking how to make the process of sliding a credit card easier?
* [E] A. Security is more important than HCI in this setting.
* [E] B. It forces us to think only in terms of the credit card itself.
* [E] C. Credit cards will soon be obsolete anyway.
* [E] D. It’s more important to stick to what people are used to than try to make small improvements.

### 2.2.18.3 David’s House (Breakfast Room)

* The problem is that we’re not focusing on the right task.
* Our task shouldn’t be how to best swipe a credit card.
* Our task should be how to most easily pay for purchases.
* <David taps his phone against the machine’s reader>
* To just do it by tapping our phones.
* But maybe that isn’t the best option.
* The important thing, though, is to focus on what we’re really trying to accomplish, not just how we’ve done it in the past.
* We can make incremental improvements to sliding a credit card, but we should always keep our eyes on the underlying tasks the user needs to accomplish.

## 2.2.19 Conclusion

### 2.2.19.1 Headshot Studio

* [C] David on left
* [A] Clips of earlier videos playing on right
* [B] Topic; Feedback cycles
* Today, we’ve talked about arguably the fundamental concept of human-computer interaction, **feedback cycles**.
* We described feedback cycles for our purposes as the exchange of input and output between a user and a system to accomplish some goal.
* [B] Topic; Feedback cycles’ ubiquity
* We discussed feedback cycles’ incredible **ubiquity** in other fields and discussions.
* [B] Topic; Gulfs of execution
* We talked about **gulfs of execution**, the distance between knowing what they want to accomplish and actually executing the steps to accomplish it.
* [B] Topic; Gulfs of evaluation
* We talked about **gulfs of evaluation**, the distance between making some change in a system and evaluating whether or not the goal was accomplished.
* [B] Topic; Seven questions for bridging gulfs
* We introduced the **seven questions** we need to ask to bridge those gulfs.
* Now that we understand these gulfs, our next goal is to understand methods for crossing them.