Once you have your design principles, you can use them as a measuring stick against the concepts you've generated to see which ones best fit. Hopefully several ideas will work within the guidelines, or could be tinkered with to fit.

But design principles can also be used from this point in the process forward to help make design decisions. When there are multiple options to choose from ("Should we ask users first, or just do it for them?"), the design principles can sometimes help make the correct decision clear.

Design principles can sometimes outlast the specific product itself, or even be extended across lines of products to give them all a similar grounding.

Summary

Brainstorming can be mysterious. Frequently an idea will come to you when you are not in a brainstorming session. Ideas seem to have a life of their own, but they can sometimes be coaxed into existence, and that's what you hope ideation will do.

The design principles you create are a way—granted, a subjective way—of measuring your ideas for value and feasibility. Of course, the only way to really tell if an idea is a good one is to play with it, test it out, and refine it. That is the topic of the next chapter.

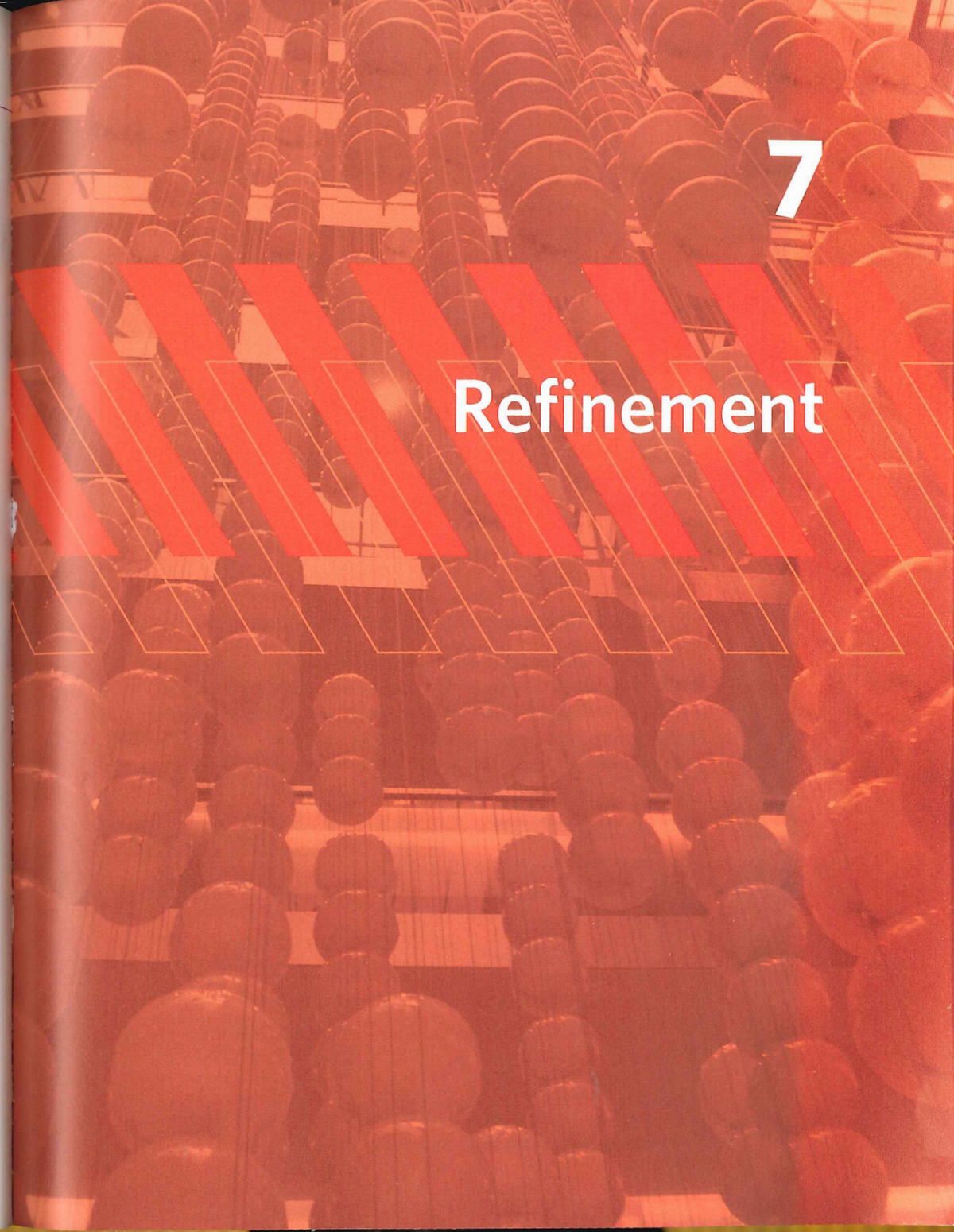
For Further Reading

Six Thinking Hats, Edward de Bono

A Technique for Producing Ideas, James Webb Young

Thinkertoys: A Handbook of Creative-Thinking Techniques, Michael Michalko

The Seeds of Innovation: Cultivating the Synergy That Fosters New Ideas, Elaine Dundon



7

Refinement

Service String

The second component of a service blueprint is the **service string**. The service string shows the big idea for the service in written and visual form, usually in the form of storyboards. Designers create service strings by putting concepts for various service moments together to form a scenario, or string, of events that provide a pathway through the service.

The service string demonstrates in a vivid way what the pathways through the service will be and provides a comprehensive, big-picture view of the new service. Viewers can see how customers order, pay for, and receive the service, and how employees provide the service. For example, a service string for the earlier car wash example would show in a single scenario customers seeing the new signs, customers using the new machine to pay for the car wash, the special washing service, the attendants who hand-dry the cars, and the new vacuum for cleaning out the cars after they are washed.

Controls

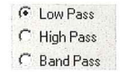
Most applications and devices that interaction designers currently design have some sort of visible controls to manipulate the features of the product—a dial to control volume on a stereo, for example, or a slider to select a date range. (The major exceptions are voice and gestural interactions, discussed later in this chapter.) **Controls** provide both the affordances needed to understand what the product is capable of, and the power to realize that capability.

This section describes many of the basic controls that interaction designers can use as a palette. Almost all of these controls have their own standard feedback mechanisms (a switch moves and stays in its new position, for instance) that interaction designers should consider:

- **Switch.** A toggle switch is a very simple control. It moves from one setting ("on") to another ("off") and stays there until changed.

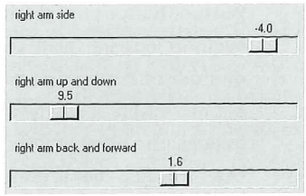
- **Button.** Buttons are the interaction designer's best friend. Once you begin to look for them, it's apparent that buttons are everywhere, all over our interfaces. In a word processing program, there are about 30 buttons visible at any given time. A mobile phone may have about 40 buttons: the number keys for dialing and a keyboard. A button is, at base, a switch that is pressed or clicked to activate it. The button can stay pressed (a **toggle button**), requiring another press to reset it (like most on/off buttons), or it can reset itself automatically (like keys on a keyboard). Buttons can be used for a wide variety of actions: from changing modes (from writing text to drawing, say) to moving an item or a cursor via arrow keys. Buttons can take many forms, from tiny icons to physical squares on a floor that can be stepped on. Buttons, however, are good only for simple actions.

- **Radio button.** Radio buttons enable users to choose from (often preset) items from a set. Typically, these are used to constrain selections, when only one answer is allowed ("What color hair do you have?" Black, Blonde, Red, Brown).



- **Dial.** Dials provide more control than buttons, allowing the user to select a setting along a continuum (such as the amount of heat on a stove's burner) or to choose between different settings or modes (such as the mode for taking pictures and the mode for viewing them on a digital camera. Dials can move freely, or simply turn from an established point to other established points on a wheel. These points are called detents. Some dials, like those often found on clothes driers, can be pushed in and pulled out, performing an action (such as turning on or off) that can vary based on the dial's rotation.

- **Latch.** A latch opens an otherwise tightly closed area. Latches are useful for keeping some areas or items hidden or safe until needed. They are good to use when a button or drop-down menu might be too easy to click or open. For example, latches are frequently used on handheld devices to keep the battery compartment safe.



- **Slider.** Sliders, like dials (although linear instead of round), are used for subtle control of a feature, often to control output (such as speaker volume) or the amount of data displayed (such as the number of houses on an interactive map). Sliders with more than one handle can be used to set a range within a range.

- **Handle.** A handle is simply a protruding part of an object that allows it to be moved or, in some cases, resized. Handles on the frames of most digital windows allow the windows to be moved around the screen or resized.

Physical-Only Controls

Some common controls are found only in the physical world and not on screens (although they can certainly manipulate objects on a screen).

- **Jog dial.** A jog dial is a type of dial that can be manipulated with a single finger, usually a thumb. It can be dial-like, or it can be a pad of buttons, typically used on small devices for moving a cursor or moving through menus. Jog dials are somewhat difficult to control, especially for young children and the elderly.

- **Joystick.** A joystick is a physical device typically used in digital gaming or in other applications that require rapid movement and intensive manipulation of remote physical or digital objects. Joysticks can move in any direction or can be constrained to move only left to right or only up and down.

- **Trackball.** A trackball is a physical device for manipulating a cursor or other digital or physical objects. Trackballs are typically in a stationary base, but the ball itself moves in any direction. A computer mouse is often a trackball in a case.

- **5-way.** A 5-way is a combination button and cursor. It generally moves a cursor on a screen in four directions (up/down, left/right) and has a button in the center in order to select what has been navigated to.

COURTESY PALM

Bill DeRouchey on Frameworks and Controls

Bill DeRouchey is a Senior Interaction Designer at Ziba Design. Bill has over 15 years of experience as a writer, information architect, product manager, coder, and interaction designer. He has designed a wide variety of products, from handheld satellite radios and medical devices to community Web sites, interactive spaces, and product architectures.

How do you go about choosing a structure or framework for your designs?

Most often, the directions that I explore are largely bounded by the physical constraints at the beginning of the client engagement. Many clients already have specific components selected for manufacturing before engaging with them, so I have to treat those as givens, specific display dimensions and resolutions being the most common example. When you're given a 160x128 pixel space to work within, that tends to inform your structure quite a bit.

Beyond that, my designs tend to follow a pattern of reminiscence. A new medical monitor needs to behave like clinicians are used to them behaving. New satellite radios need to convey reminiscent qualities of "radio" so that people have a basis from which they approach the device. It's all about giving people a head start for understanding how to interact with the new product in front of them. This allows it to better fit into their lives as seamlessly as possible.

You've written a lot about the history of the button. Why is that important?

Interaction design existed as an activity decades before it was explicitly named as one. Industrial designers applied knobs, switches, and buttons to their products, and determined how they would be used by consumers. So these products created a rich history of people interacting with technology long before computers entered our daily lives. These first decades of products paved the way and formed our expectations of interacting with products.

This is where the button gets interesting. Consider that our main concern as interaction designers today is how we interact with products. In the early days, the question was why we should interact with products at all? Convenience, luxury, efficiency and visions of the leisurely future were all used as aspirational triggers to buy blenders, washers, radios, and more. And all of this aspiration was communicated via imagery of fingers pushing buttons.

Bill DeRouchey on Frameworks and Controls *(continued)*

The phrase "push-button" itself meant easy, simple, even-you-can-use-this product. That's a lot of social burden placed on a single UI widget, which is why I love this story.

What should interaction designers know about controls?

Physical controls have strong metaphors and history attached to them. Knobs and sliders typically indicate that you're looking for something vague along a spectrum: the right volume or temperature setting. Buttons and switches typically indicate a choice is being made. Turn the lights on. Start the microwave. Controls usually do only one thing.

Accordingly, one of the biggest challenges of controls is that space, size, and cost limit you for how many features are important enough to warrant their own physical controls. Do you really need to adjust bass levels that often, or do you bury that feature in another control somehow? Like all design, it's a delicate dance to determine this hierarchy, and the best way to solve it is to put prototypes in front of other people.

What are the most important things to remember when laying out controls?

Laying controls requires a strong sense of hierarchy, zoning, and priority. Control panels typically focus on a tight set of tasks, with one hero task in that set. In air conditioners, changing temperature is more important than adjusting schedules. In radios, adjusting volume is the hero task. These controls should be larger, offset, or otherwise designed to have the highest priority. It should be clear to people what is the single most important thing to do. Determine your hero task.

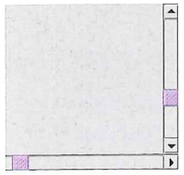
A common mistake is designing all the controls with an overly uniform look and feel. It may look clean to have 12 different buttons with uniform shape and color lined up into a grid, but that approach offers no quick visual appraisal to determine what control does what. In these situations, the labels become more important, creating a secondary problem.

Digital-Only Controls

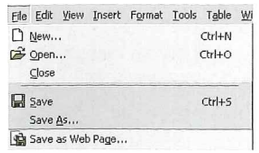
While many controls are found in both the physical, analog world and the digital one, some controls are only found on screens. These digital controls have grown from the original graphical user interface (GUI) vocabulary that was invented at Xerox PARC in the 1970s, reinvented in the 1980s in the Macintosh and PC operating systems, and added to and expanded by Web conventions in the 1990s:

- **Check box**. A check box enables users to select items from a short list.

- **Twist.** Twists turn up or down, either revealing or hiding content or a menu in a panel.



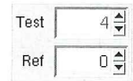
- **Scroll bar.** Scroll bars enable users to move content within a particular window or panel. Scroll bars can be vertical or horizontal. Scroll bars themselves can be manipulated via the cursor or buttons (for instance, by using arrow keys).
- **Drop-down menu.** Drop-down menus allow designers to cluster navigation, functionality, or content together without having to display it all at once. Drop-down menus can be displayed by rolling over them, or they can be opened with a click. They can retract after a selection has been made or the cursor rolls off them, though not necessarily.



- **Multiple-selection list (or list box).** Multiple-selection lists enable users to select multiple items in a list.

- **Text box**. Text boxes enable users to enter numbers, letters, or symbols. They can be as small as (and constrained to) a single character or as large as the whole screen.

- **Spin box.** Spin boxes are text boxes with additional controls that enable users to manipulate what is inside the text box without having to type a value. They are good for suggesting values in what otherwise might be an ambiguous text box.



The combination of one (and usually more) controls plus the system response is called a **widget**. Widgets are the building blocks of any application or device. An MP3 player, for instance, is made of widgets: one for controlling volume, one for controlling the playing of music files, one for organizing files, one for exporting files, and so on. In each case, the user uses controls to perform an action, and the system responds. All applications and devices are made up of widgets.

Non-traditional Inputs

We are arriving at a time when keyboards, mice, and styluses aren't the only—and possibly not even the primary—way we interact with the digital world. With the dawn of ubiquitous computing, interactive environments, and sensor-enabled devices (see Chapter 9), people will engage with many different sorts of objects that have microprocessors and sensors built into them, from rooms to appliances to bicycles.

The controls for these faceless interfaces are the human body: our voices, our movements, and simply our presence.

COURTESY KERRY BODINE

Figure 7.13

The author screams at Kelly Dobson's Blendie, a voice-controlled blender, to get it to frappé.

Voice

Widespread implementation of **voice**-controlled systems has been on the horizon for at least a decade now. For now, voice-controlled interfaces are most prevalent (naturally) on phone systems and mobile phones. For example, people call their banks and perform transactions or dial their mobile phones with just their voices. Voice commands typically control limited functionality, and the device typically has to be ready to receive voice commands, either because it only functions via voice commands (as with automated phone systems and some voice-controlled devices—see **Figure 7.13**), or because it has been prepared to receive voice commands, as with mobile phones that allow voice-dialing.

Gestures

To most computers and devices, people consist of two things: hands and eyes. The rest of the human body is ignored. But as our devices gain more awareness of the movement of the human body through sensors such as cameras, the better able they will be to respond to the complete human body, including **gestures**. Devices like the Wii and the iPhone with their built-in accelerometers allow for all manner of new ways of controlling our devices via movements in space. See **Figure 7.14**.

COURTESY CANESTA AND KICKER STUDIO

Figure 7.14

This gestural entertainment center uses a camera from Canesta to detect gestures in space that control the television.

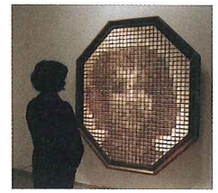
Designers need to be especially aware of several issues when designing gestural interfaces:

- **Physiology and kinesiology.** Designers have to know how humans move and what the limitations are for that movement. For example, holding an arm out and making gestures can be quickly tiring—a condition known as "gorilla arm."
- **Presence and instruction.** Since there might be no visible interface—for example, consider the hands-free paper towel dispenser in many public restrooms—letting users know a gestural device is there and how to use it needs to be addressed.
- **Avoiding "false positives."** Since human beings make gestures all the time in the course of just moving around, designing and then detecting deliberate gestures can be challenging.

- **Matching gesture to task.** Without standard controls, figuring out the best motion to trigger an action is important. Simple gestures should be matched to simple tasks.

Presence

Some systems respond simply to a person's **presence**. Many interactive games and installations such as Daniel Rozin's "Wooden Mirror" (**Figure 7.15**) respond to a body's being near their sensors.



COURTESY DANIEL ROZIN

Figure 7.15

The "Wooden Mirror" creates the image of what is in front of it (seen by a camera) by flipping wooden blocks within its frame.

There are many design decisions to be made with presence-activated systems. Consider a room with sensors and environmental controls, for example. Does the system respond immediately when someone enters the room, turning on lights and climate-control systems, or does it pause for a few moments, in case someone was just passing through?

In addition, sometimes users may not want to be known to be present. Users may not want their activities and location known for any number of reasons, including personal safety and simple privacy. Designers will have to determine how and when a user can become "invisible" to presence-activated systems.

Summary

Refinement of design concepts is about making smart, deliberate choices about how the concept would work and could be built given the known constraints. It's about using the known laws of interaction design to guide design choices, and about putting in the right affordances and feedback so that users can create the right mental model of the product in order to properly use it.

Of course, right now, these are just documents; they don't live and breathe and you cannot really "interact" with them. For that, prototyping is necessary, and that is what the next chapter covers.

For Further Reading

About Face 3: The Essentials of Interaction Design, Alan Cooper, Robert Reimann, and David Cronin

The Design of Everyday Things, Donald A. Norman