Building User Interfaces

Design Paradigms, Patterns, & Languages

Professor Yuhang Zhao

Logistics

— Midterm: Wednesday (alternative), Friday (main)
— 44-point Build, 40-point Design, 16-Integrate
— Prior quizzes

What we will learn today?

— Design paradigms
— Design patterns
— Design languages

Recap: What is interaction design?

Interaction Design

Definition: Defining behaviors for a system that engages the full spectrum of its user's perception, cognition, and movements.

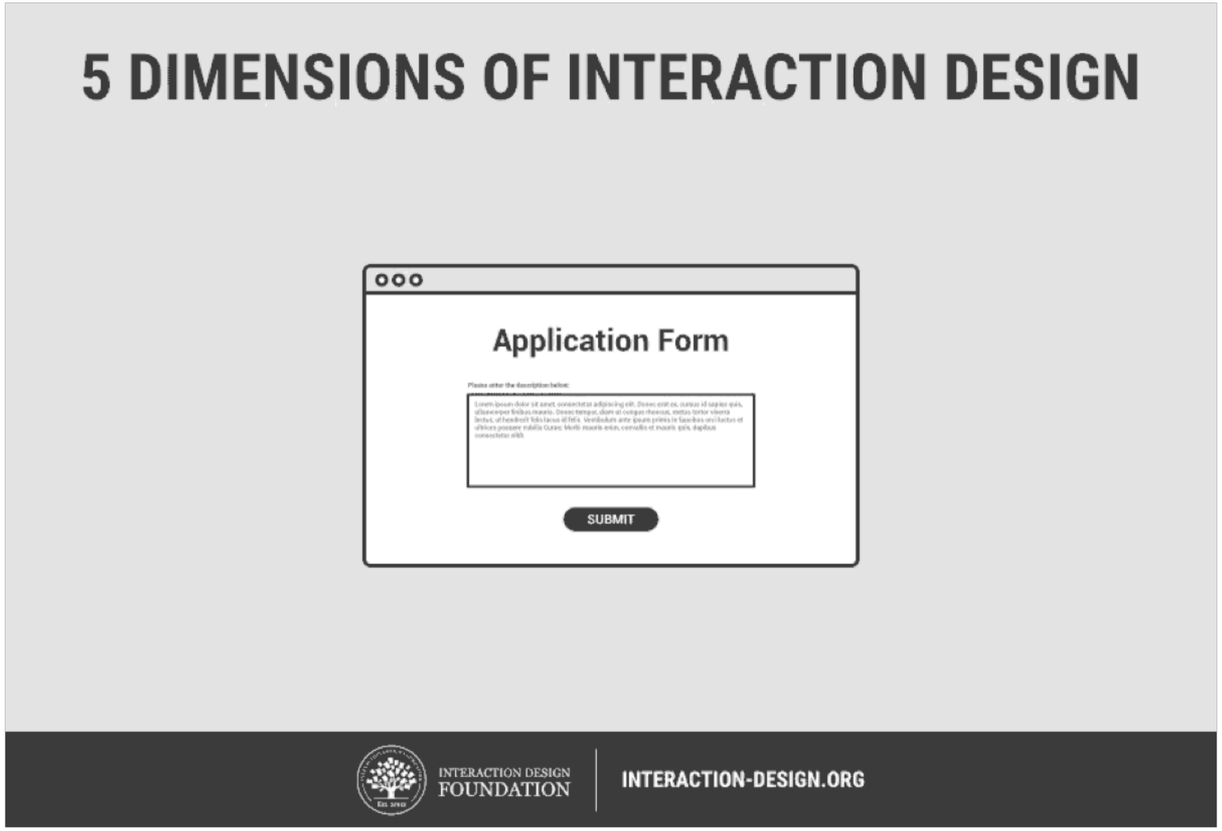
Differs from visual design in its closer and more complex relationship to user behavior and context.

Example: visual designers do not think about navigation models!

Five Dimensions of Interaction Design[1]

1. **1D**: Words
2. **2D**: Visual representations
3. **3D**: Physical objects and space
4. **4D**: Time
5. **5D**: Behavior

We talked about *visual design* and *navigation*, but how do we address all these dimensions?



[1] Interaction Design Foundation

Interaction Design Paradigms

What is a Design Paradigm?

Definition: An archetypal solution or an approach to solving design problems.

Historical Interaction Design Paradigms [9]

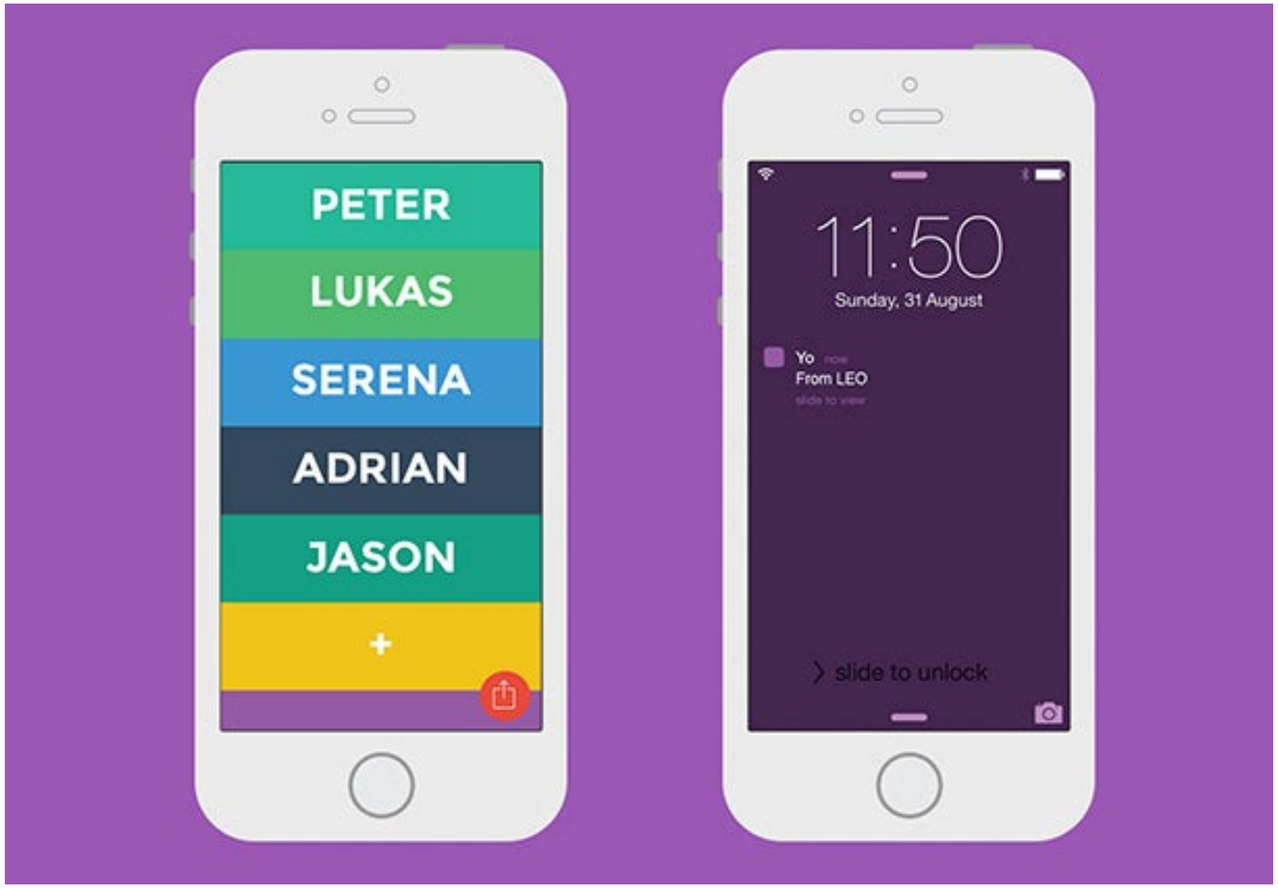
1. Implementation-centric
2. Metaphoric
3. Idiomatic

[9] Cooper et al., 2014, About Face

Implementation-centric Design

Definition: Interaction design maps directly to how system functions are implemented.

Source[2] [3]



[2] Pintrest

[3] Entrepreneur Magazine

Pros & Cons of Implementation-centric Design

Pros:

1. Very easy to build, easy to debug, easy to troubleshoot

Cons:

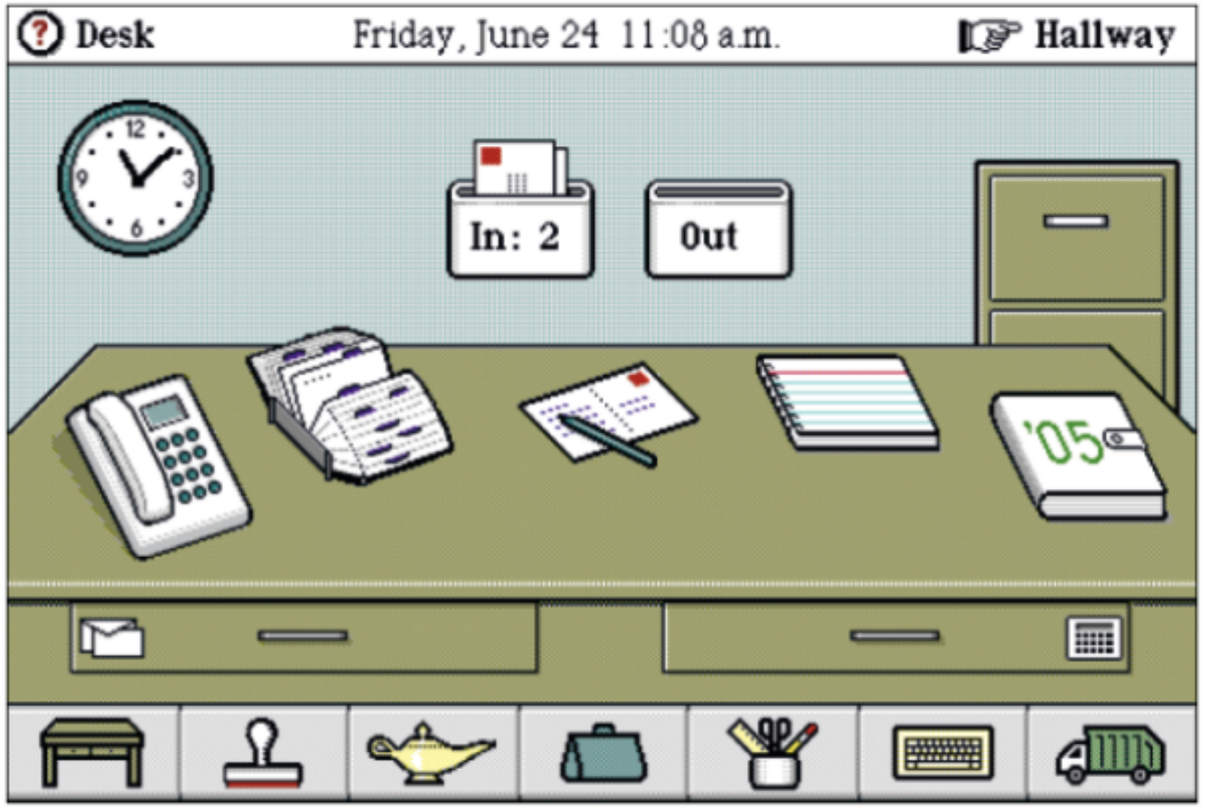
1. Requires learning how the functions work
2. Requires skills in using the functions
3. The system cannot perform high-level actions

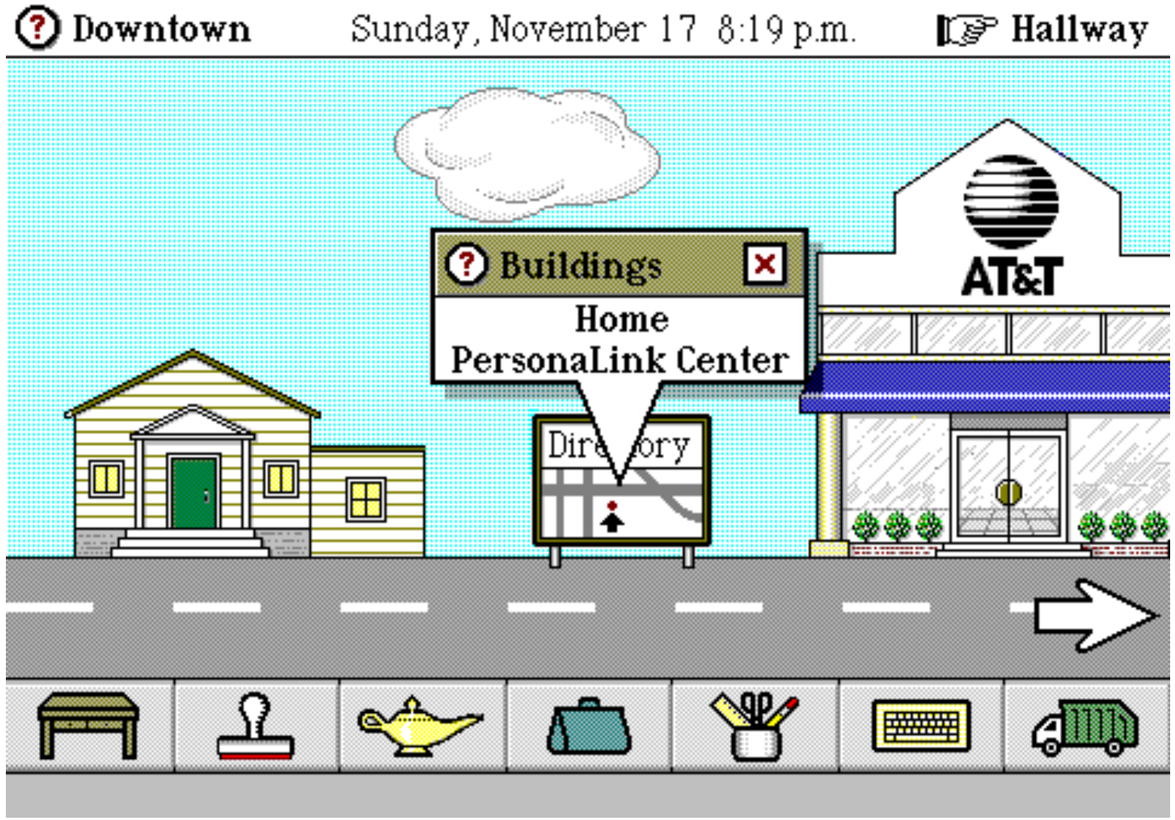
Metaphorical Design

Definition: Following a real-world metaphor that users are expected to be familiar with.

Metaphorical designs "jump-start" user mental models, rely on their existing knowledge of how things work in the real-world, and thus eliminate learning.

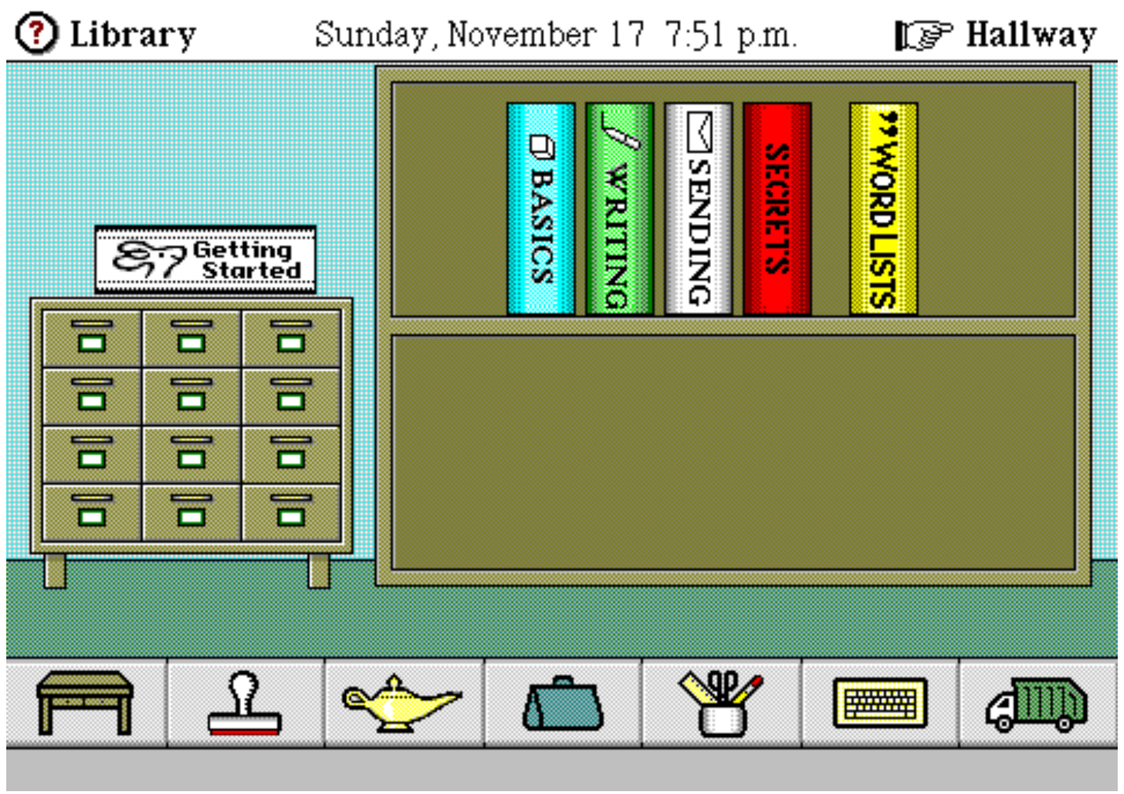
Source[4]

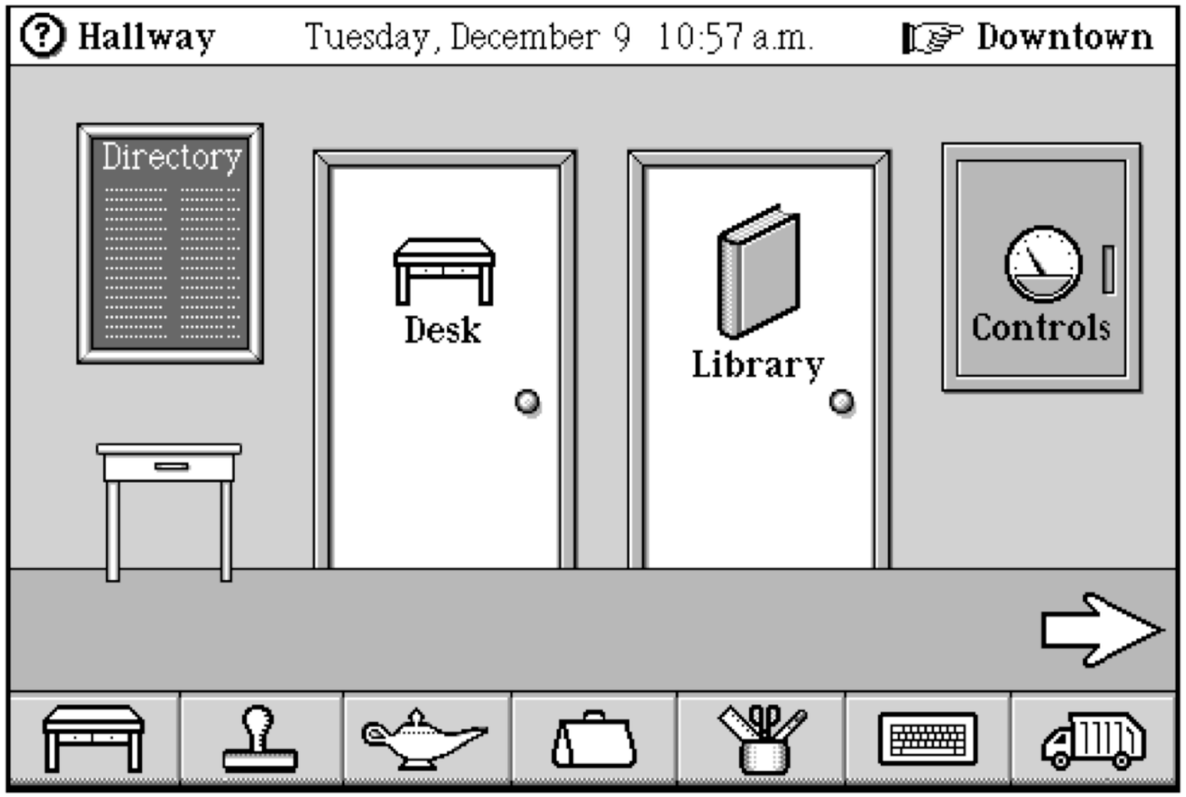




[4] Wikipedia: Magic Cap

Source[56]





[5] Wikipedia: Magic Cap

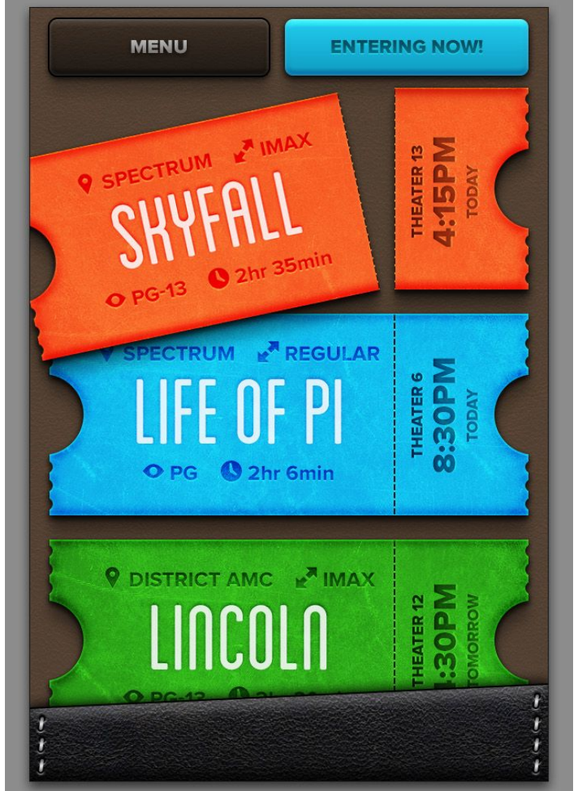
[6] NN Group: The Anti-Mac Interface

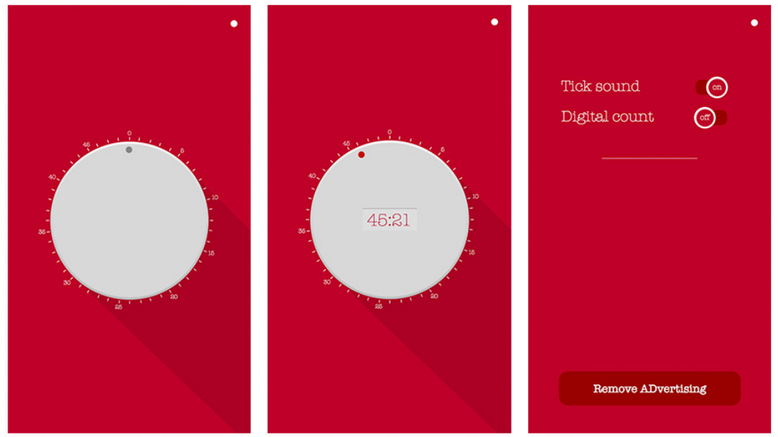
Source[7]

[7] UX Planet: Metaphorical Design

Source[8]

[8] Apple App Store: 76 Synthesizer





AND REDESIGN FOR APPLE WATCH

Pro Tip 1: Metaphors may use a familiar model from another domain (*e.g.*, building vs. computer windows); *analogies* are similar to models in the same category (*e.g.*, physical cards vs. e-cards).

Pro Tip 2: Metaphors can be applied at different levels of abstraction.

Pro Tip 3: Mixed metaphors bring together models from different domains in a single design.

Global Metaphor

Definition: A *global metaphor* provides a single, overarching framework for all the metaphors in the system (*e.g.*, Magic Cap).

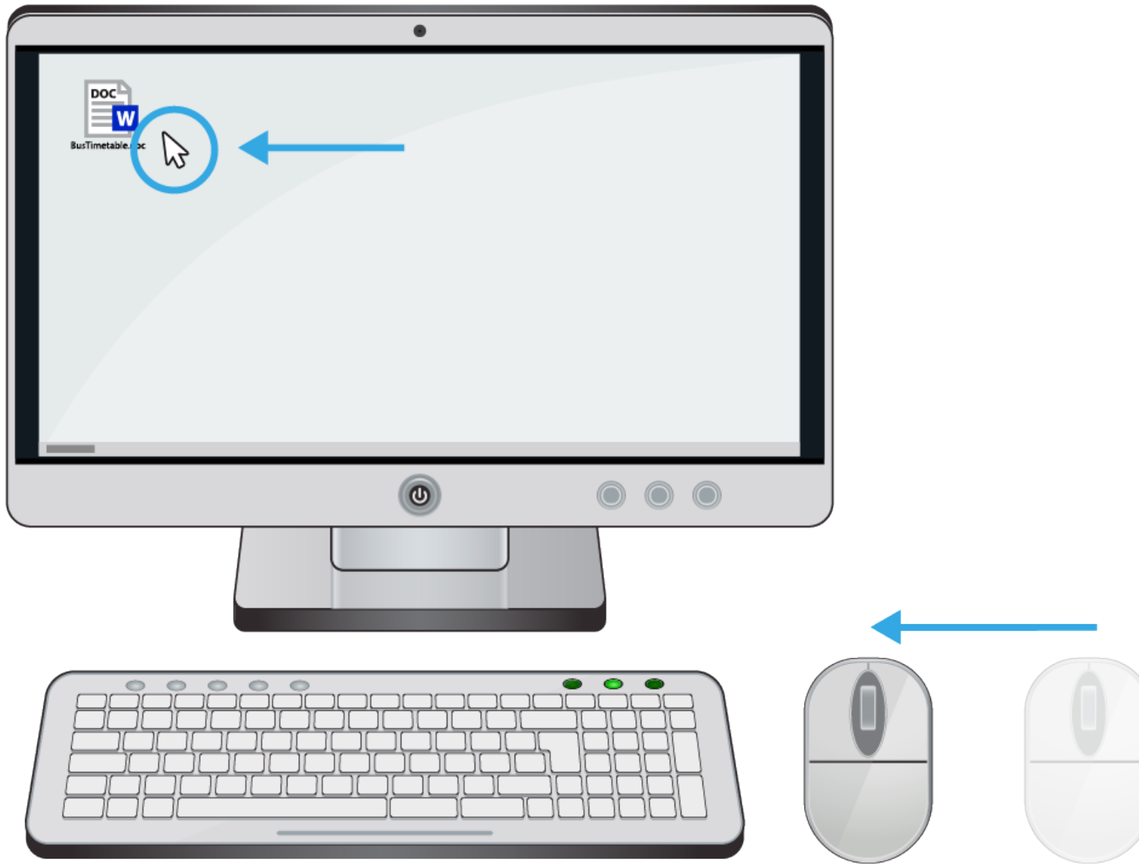
Pros: They work well in expert interfaces where the interface simulates a real-world system.

Cons: Inability to scale; lack of familiar real-world system for entirely new capabilities; cultural differences; inability to adapt as capabilities evolve.

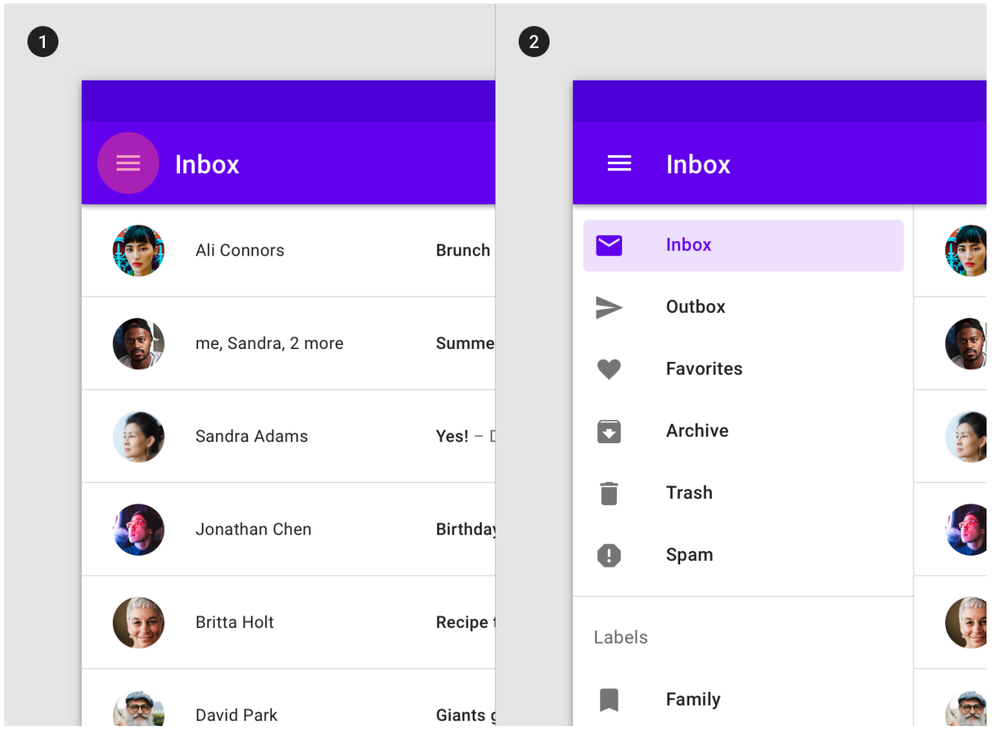
Idiomatic Design[10]

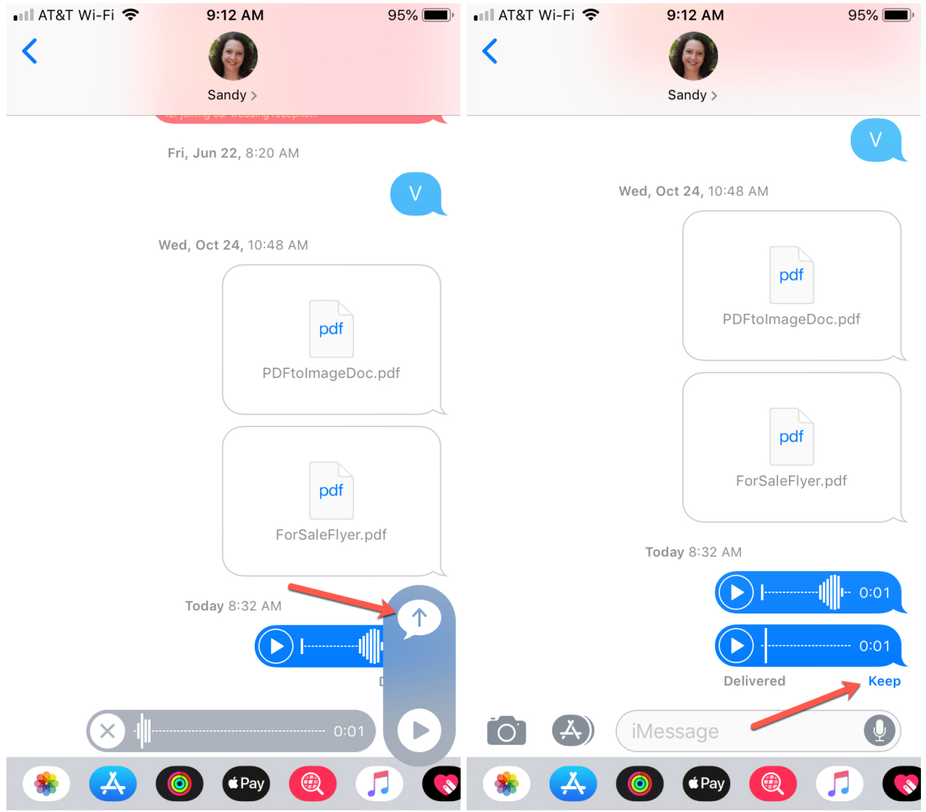
Definition: Building dedicated, highly expressive interaction capabilities that users must learn.

Mapping cursor movements on a screen to mouse movements is an extremely successful example.



[10] Image Source





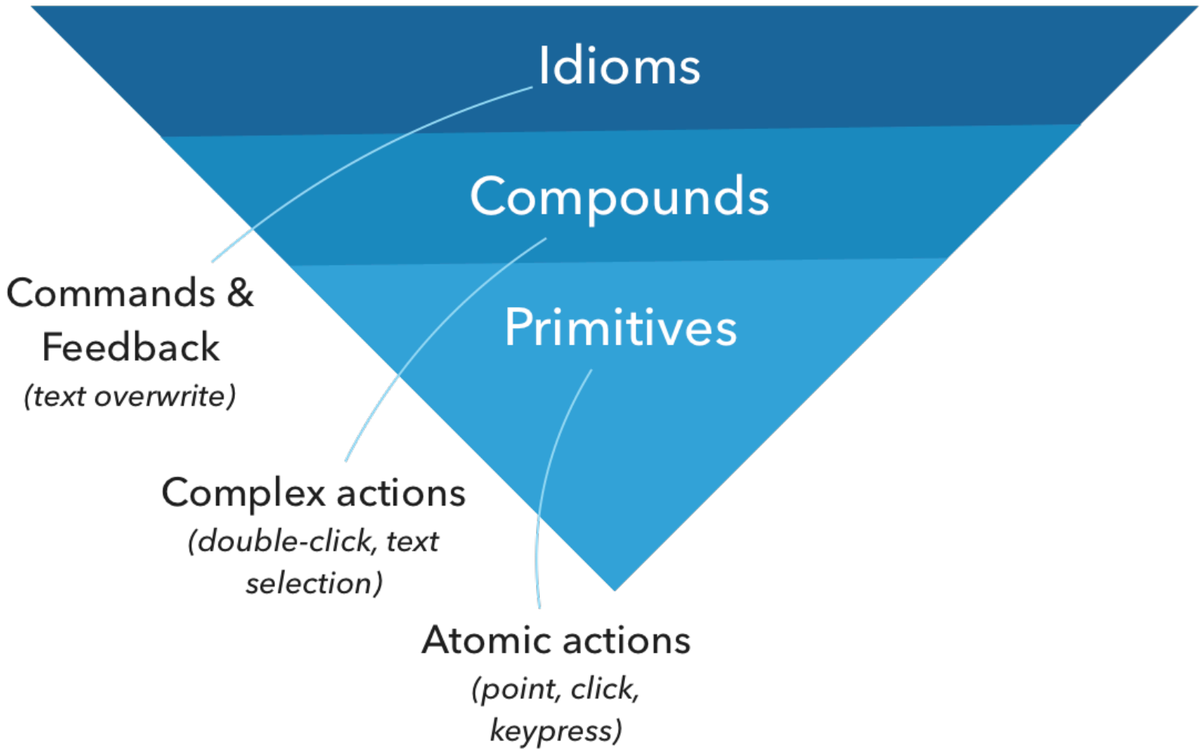
[11] Image Source

[12] Image Source

Developing Idioms[13]

In designing idioms involve, three elements are established:

1. **Primitives**: atomic actions, e.g., point, click
2. **Compounds**: complex actions, e.g., double-click
3. **Idioms**: higher-level elements, e.g., deleting text



[13] Cooper et al., 2014, About Face

Affordances

Affordances

Definition: The perceived properties of a design element that give clues about how to interact with it. Designers have borrowed the concept from ecological psychology.

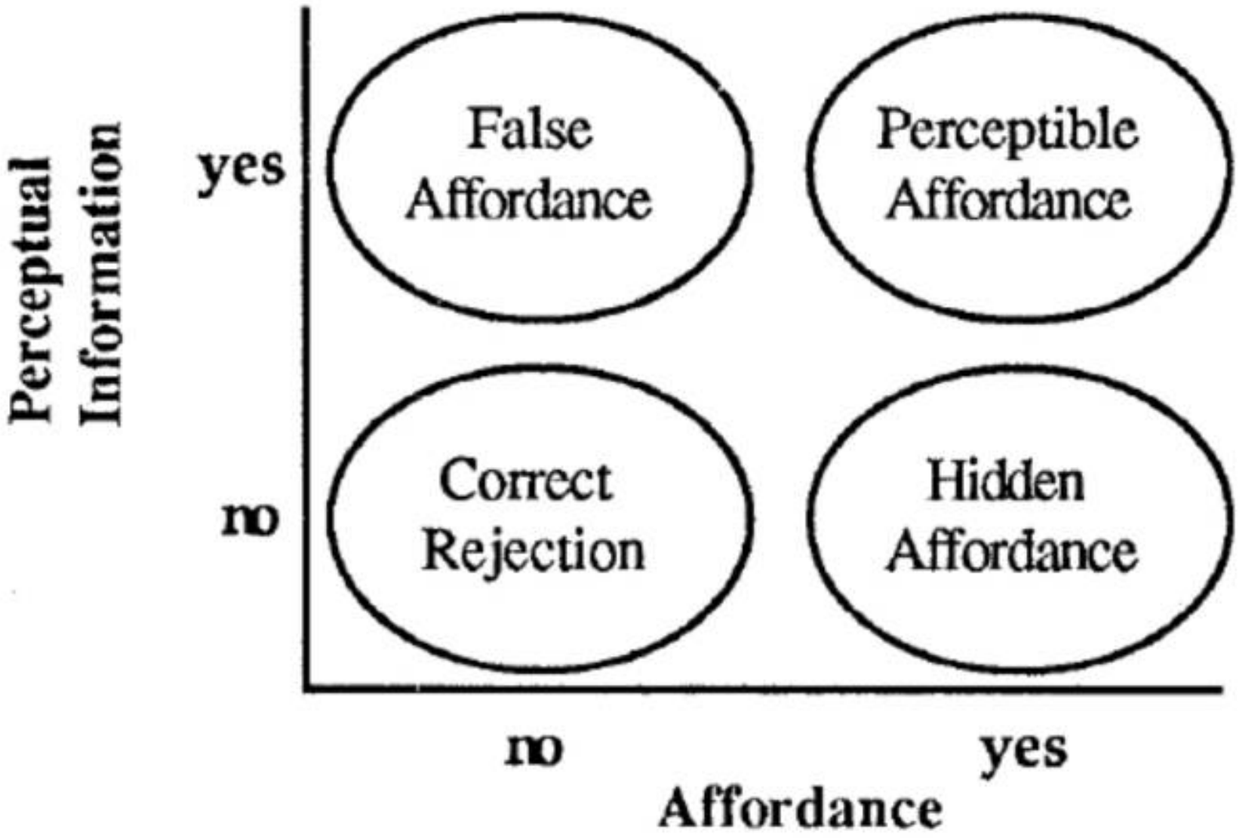
Theoretical Roots: James Gibson (1977, 1979) suggested that the human environment is structured in a way that communicates action possibilities through *affordances*.

Which environment affords *walking*?

Affordances in Design

Perceptible affordances enable users to intuitively recognize actions that are possible with interface elements.[14]

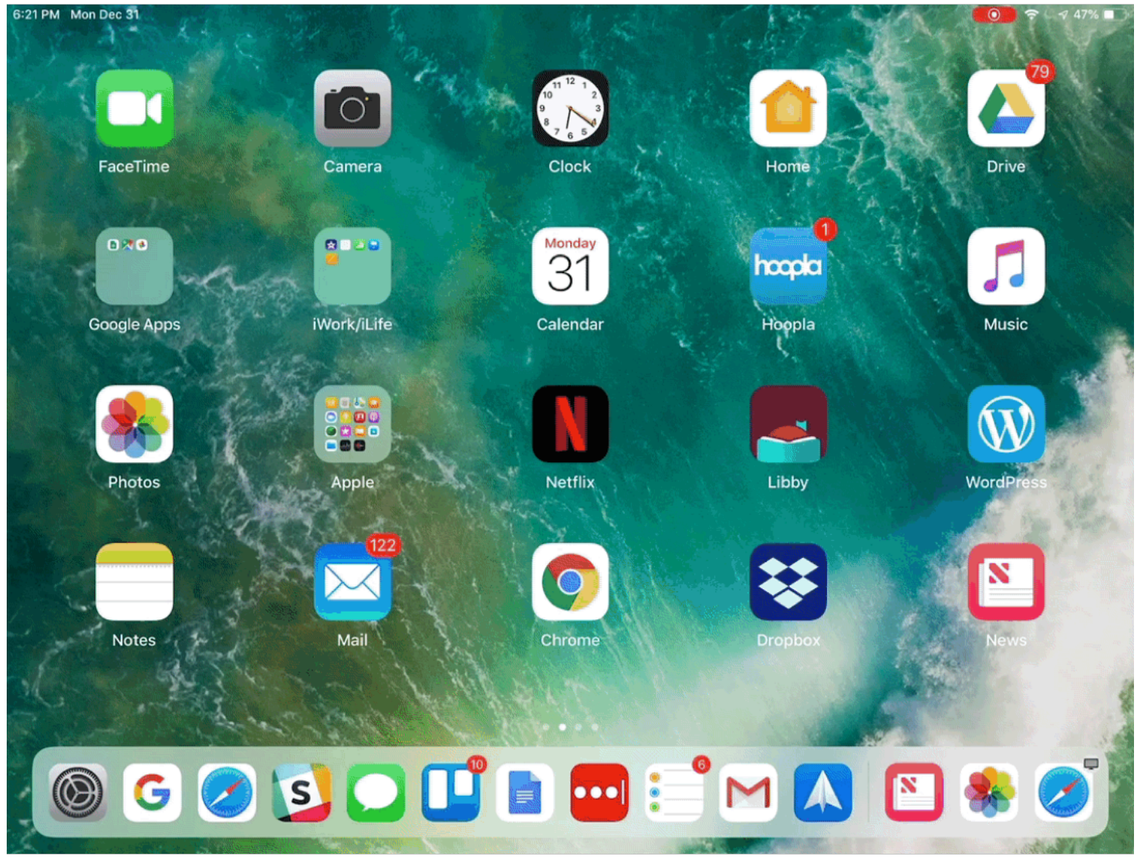
Affordances can also be *hidden* and *false*.



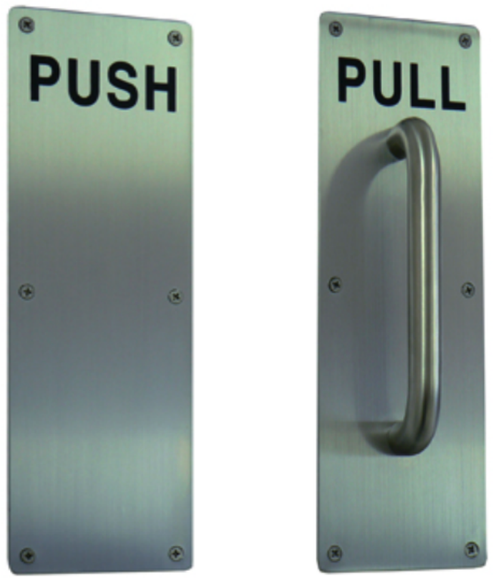
[14] Figure: Gaver, 1991, *Technology Affordances*

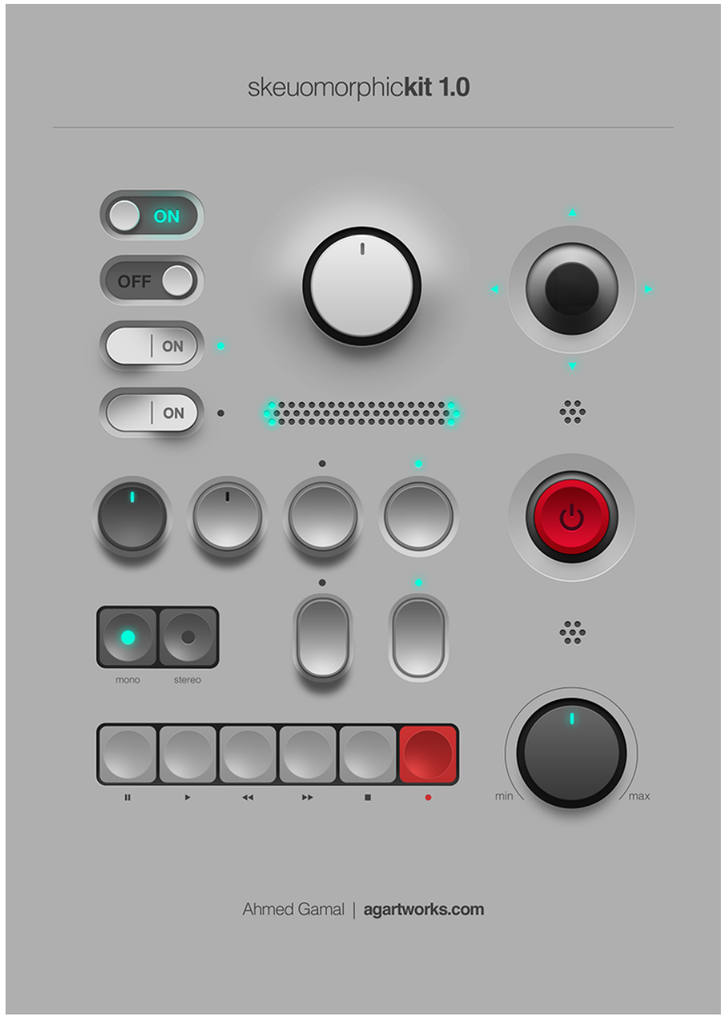
False Affordances: an action that is perceived by the user but in fact does not work as expected.

Hidden Affordance: the affordance is not too obvious.



Perceptible Affordances: an object's characteristics imply an action.





In-Class Activity

Metaphor & Affordance Deconstruction

Angle PRQ

$$\cos A = \frac{(9\sqrt{2})^2 + 13^2 - 5^2}{2(9\sqrt{2})(13)}$$

$\cos A = 0.924$

$A = 22.38$

6. $f(x) = x^2 + 8ax + 4a^2$, $x \geq 0$

$g(x) = 6x - 2a$, $x \in \mathbb{R}$

(a) $f(g(x)) = f(6x - 2a)$

$= (6x - 2a)^2 + 8a(6x - 2a) + 4a^2$

$= [36x - 12ax - 12ax + 4a^2]$

$= [36x - 24ax + 4a^2] + 48ax - 16a + 4a^2$

$= 36x^2 + 24ax - 8a^2$

IF $a = 4$

(b) $f(g[2])$

$= f(6(2) - 2(4))$

$= f(4)$

$= 4^2 + 8(4)(4) + 4(4)^2$

$= 208$

(c) $f^{-1}(x)$: $x^2 + 8(4)x + 4(4)^2$

$y = x^2 + 8(4)x + 4(4)^2$

$y = x + 32x + 64$

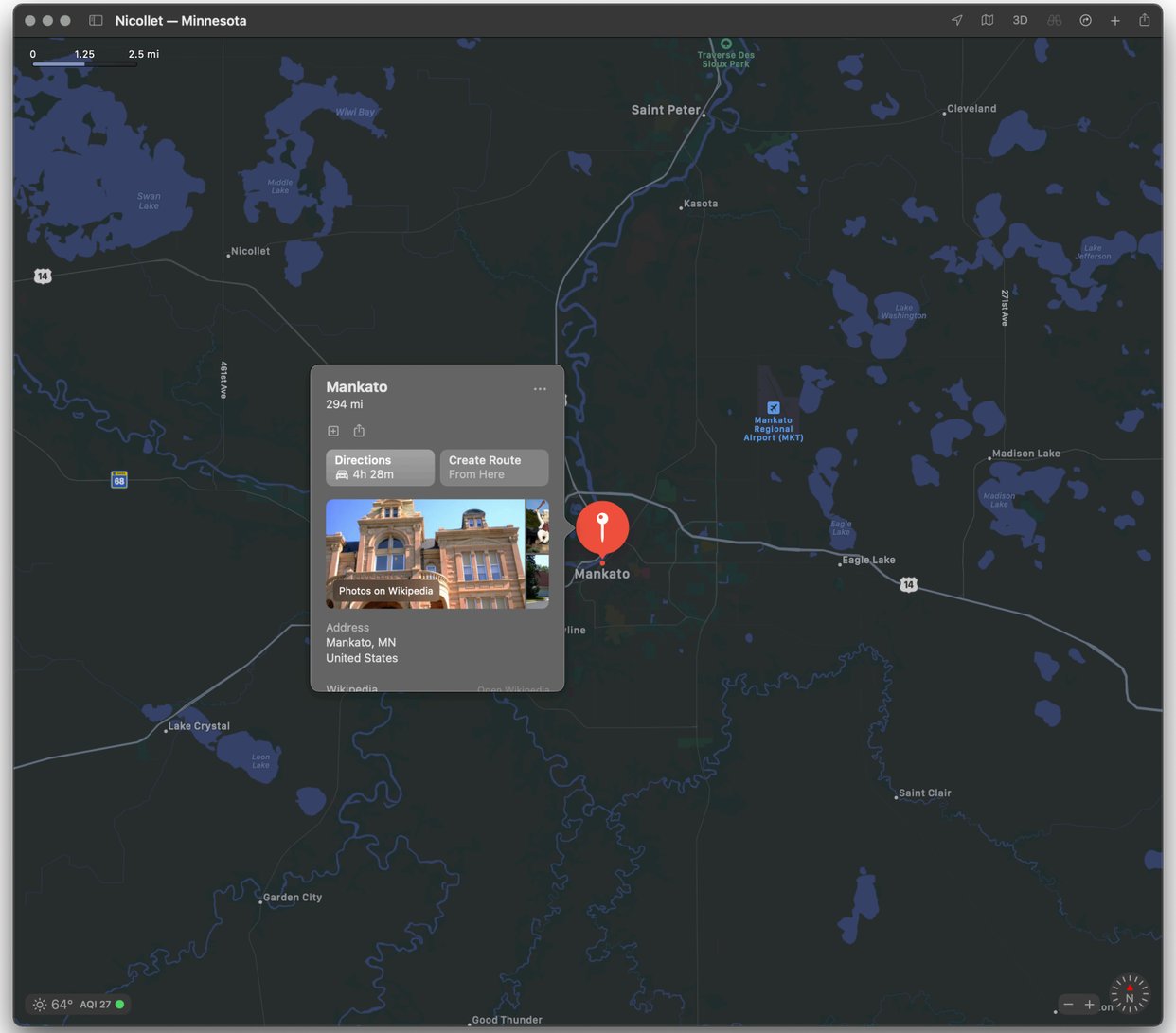
$y = (x + 16)^2 - 192$

$y + 192 = (x + 16)^2$

$\sqrt{y + 192} = x + 16$

$-16 \mp \sqrt{y + 192} = x$

So $f(x) = -16 + \sqrt{x + 192}$


Nicollet — Minnesota
3D
0
1.25
2.5 mi
Traverse Des Sioux Park
Saint Peter
Cleveland
Wiwi Bay
Swan Lake
Middle Lake
Kasota
Nicollet
14
Lake Jefferson
Lake Washington
271st Ave
461st Ave
Mankato
294 mi
Directions
4h 28m
Create Route
From Here
Photos on Wikipedia
Address
Mankato, MN
United States
Wikipedia
Mankato Regional Airport (MKT)
Madison Lake
Eagle Lake
Mankato
Eagle Lake
14
68
Lake Crystal
Loon Lake
Saint Clair
Garden City
Good Thunder
64°
AQI 27

Design Patterns

Design Patterns

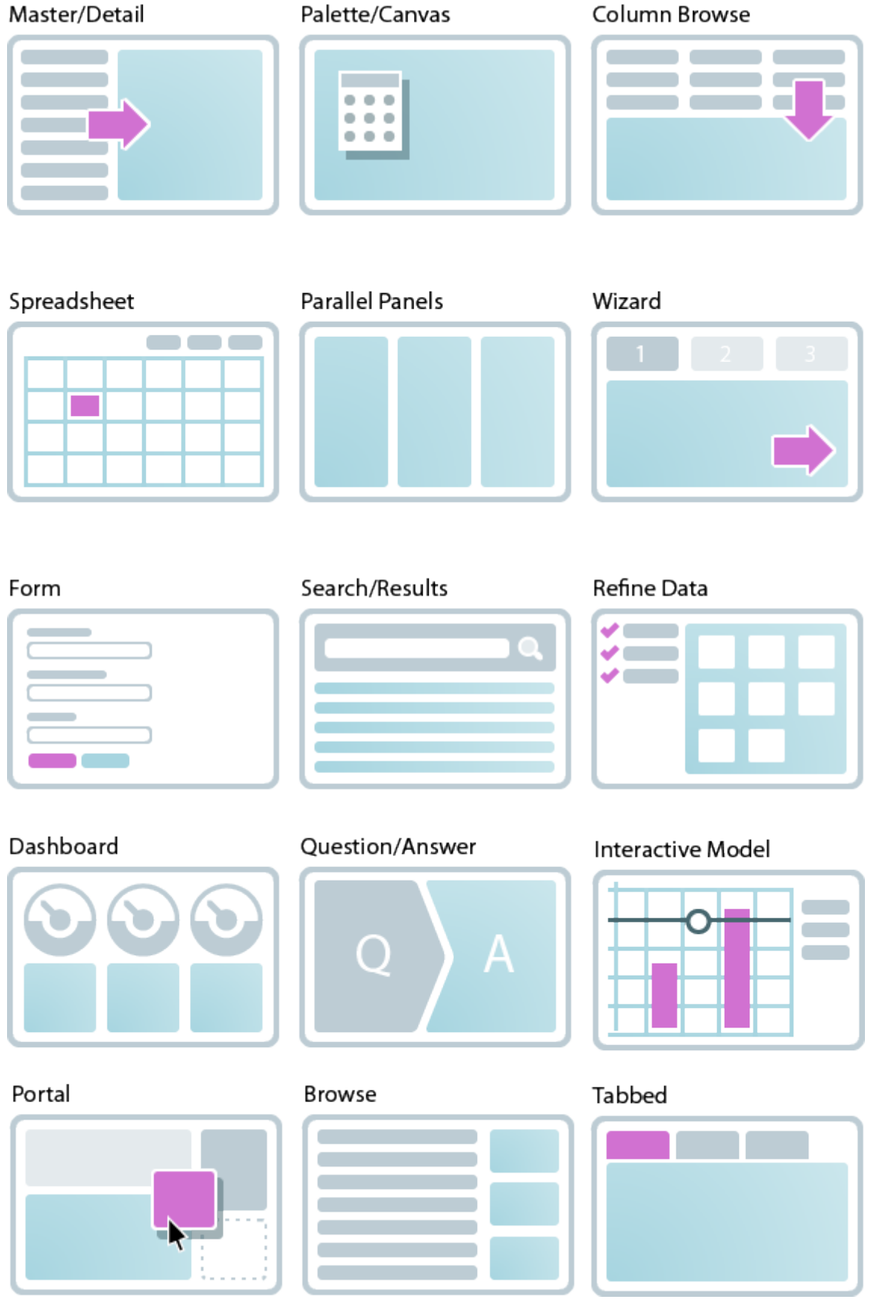
Definition: A design pattern is a general, reusable solution to a commonly occurring problem within a given context.

Originally developed by Christopher Alexander (1977; *A Pattern Language*) to address problems in architecture and city planning.[15]

[15] Smart Cities Dive

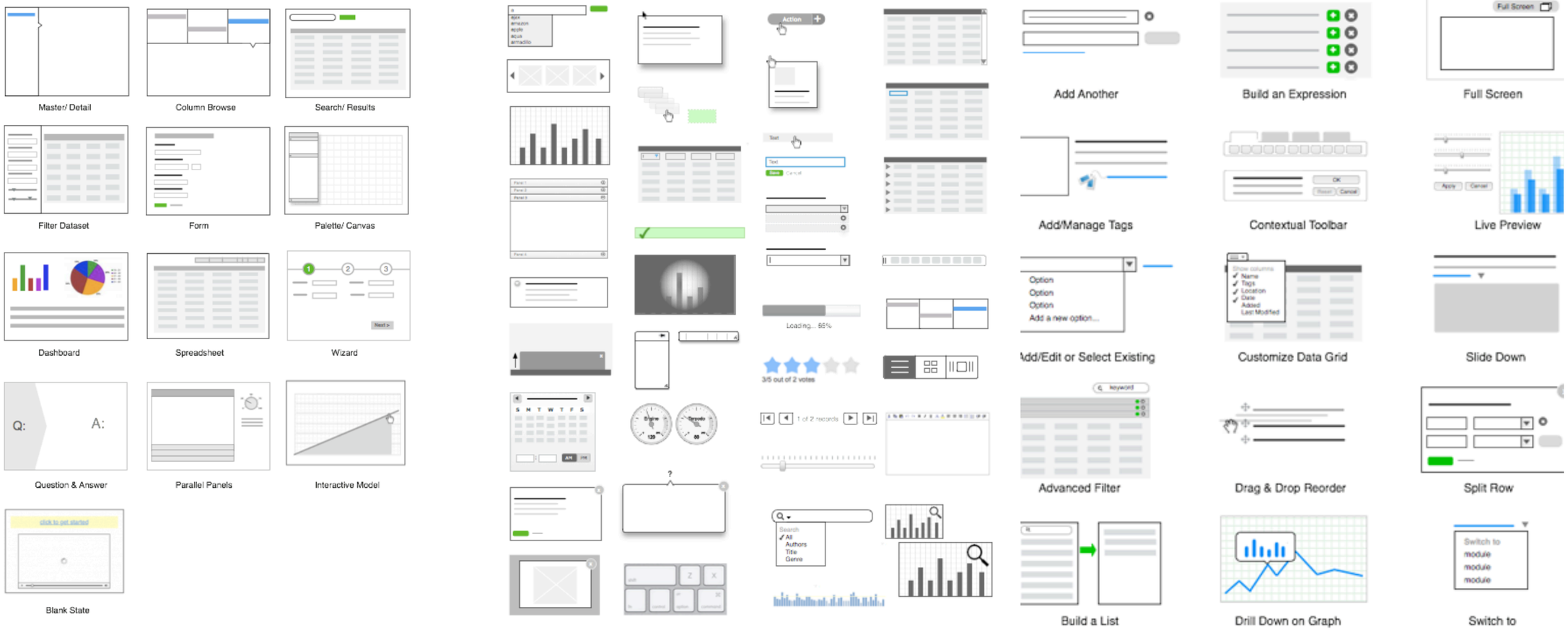
Design Patterns in UX

In the last decade, designers have also developed and refined patterns for overall structure and organization, components and controls.[16]



[16] Neil, 2010, 12 Standard Screen Patterns

Source[17]



[17] Neil, 2010, 12 Standard Screen Patterns

Pros & Cons of Design Patterns

Pros:

1. Reducing design time and effort
2. Improving the quality of design solutions
3. Establishing familiarity across systems
4. Providing a baseline or state of the art

Cons:

1. Not every design problem will warrant a pattern
2. Patterns may not exist for new design spaces

Design Languages

The Problem with Patterns

Problem 1. Can I piece together different patterns to make a complete design? **No**, as this eclectic design would lack coherence.

Problem 2. How do I choose which pattern to use? Are patterns interchangeable? **No**, there has to be a *principle* to the selection of patterns.

Problem 3: Pattern languages help you create a design that is consistent vertically. How do we create a system that is consistent *horizontally*? I.e., how do we achieve visual and behavioral consistency in designs?

The solution: Design languages!

Enter Pattern Languages

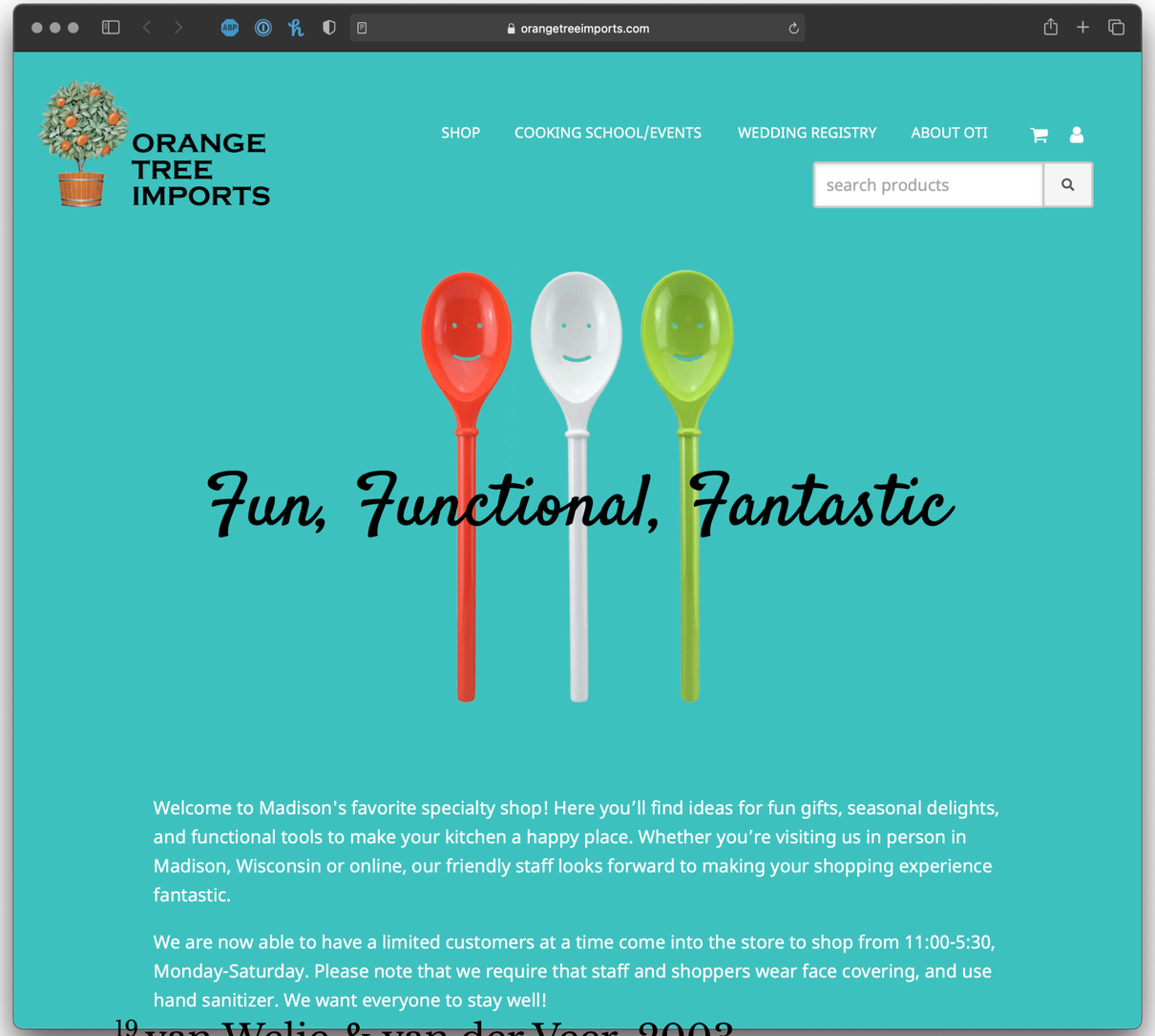
Define: A complete and hierarchical collection of patterns for a family of design problems.

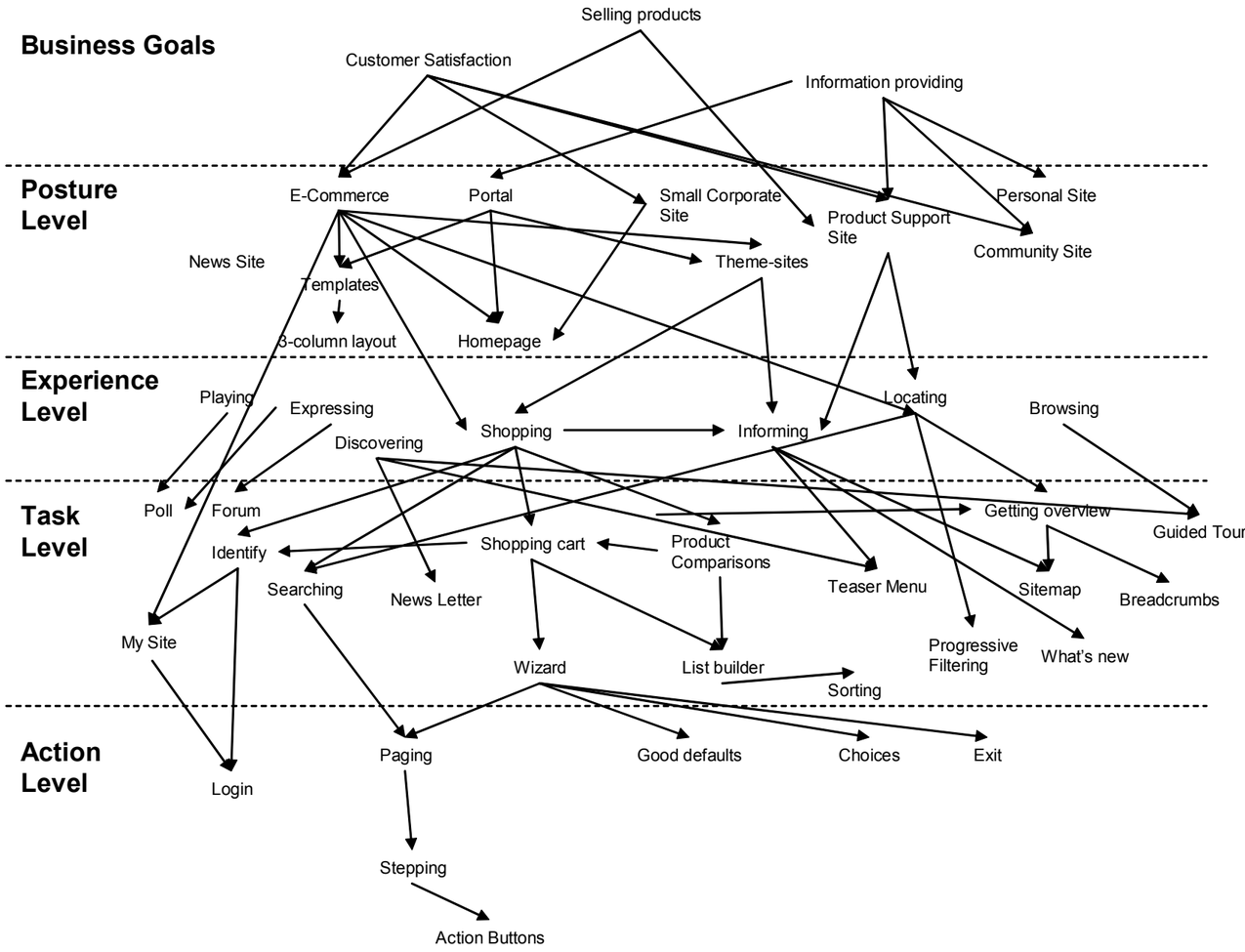
Patterns are *words* (e.g., a component) that are connected with grammar rules to make *sentences* (e.g., a screen) and eventually *language* (e.g., user experience).[18]

The pattern language can be thought of as patterns being applied at different *levels*. Let's see an example.

[18] Kruschitz & Hitz, 2009

Source[19]





[19] van Welie & van der Veer, 2003

Posture-Level Patterns

Definition: The *structure* that an application follows, i.e., what *type* of application it is, e.g., "an e-commerce app," "a social media app," or "a personal homepage site."

Elements of a Posture-level Pattern

Once we determine the posture of an application, it gives us guidance on:

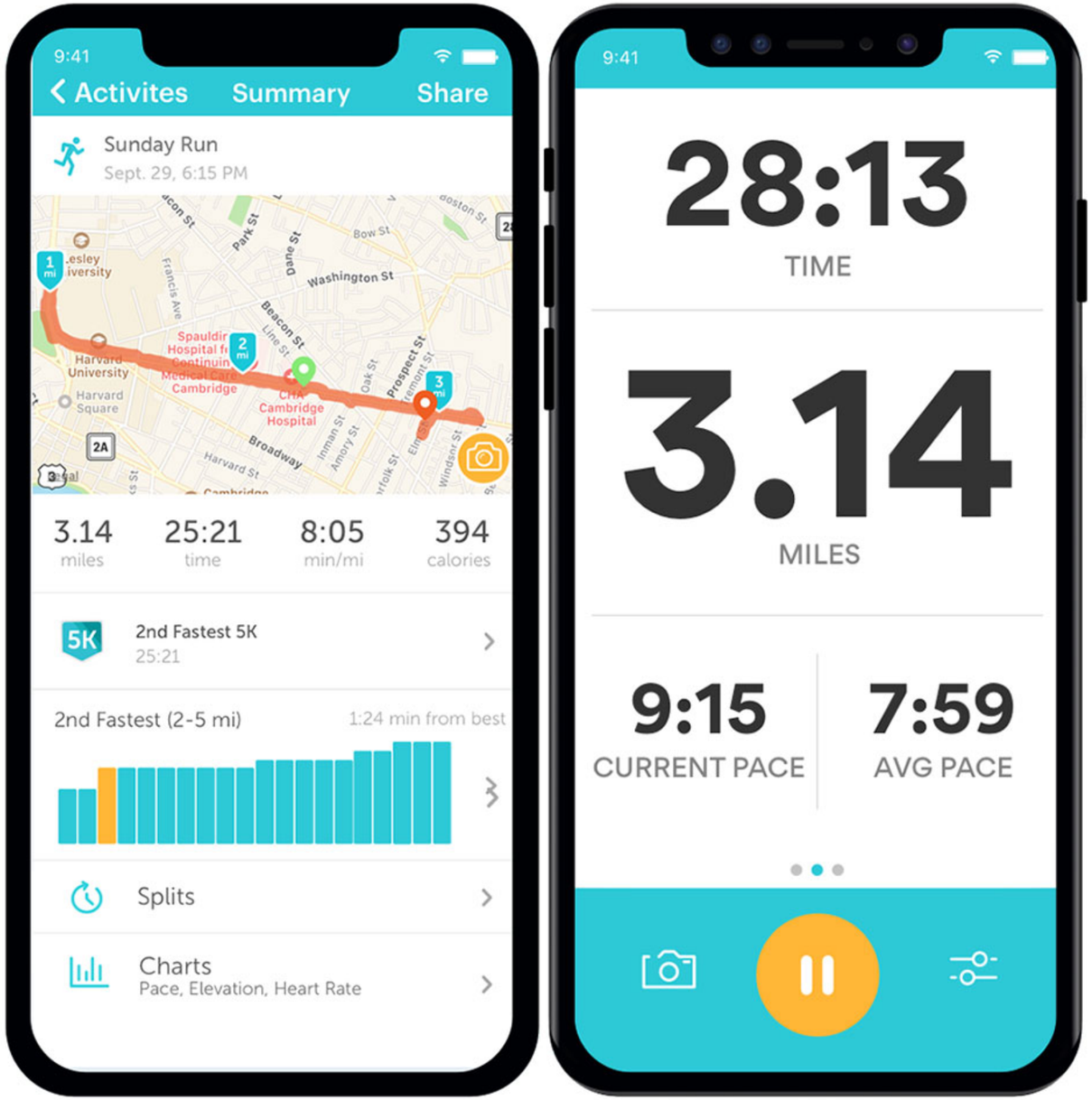
— Structure
— Components
— User experience
— Alternatives/competitors

Structure: Central canvas with supporting panels[22]

Components: Canvas, dashboard, score panel, data summary

UX: Measurement during the activity, review later

Competitors: Strava, RunKeeper



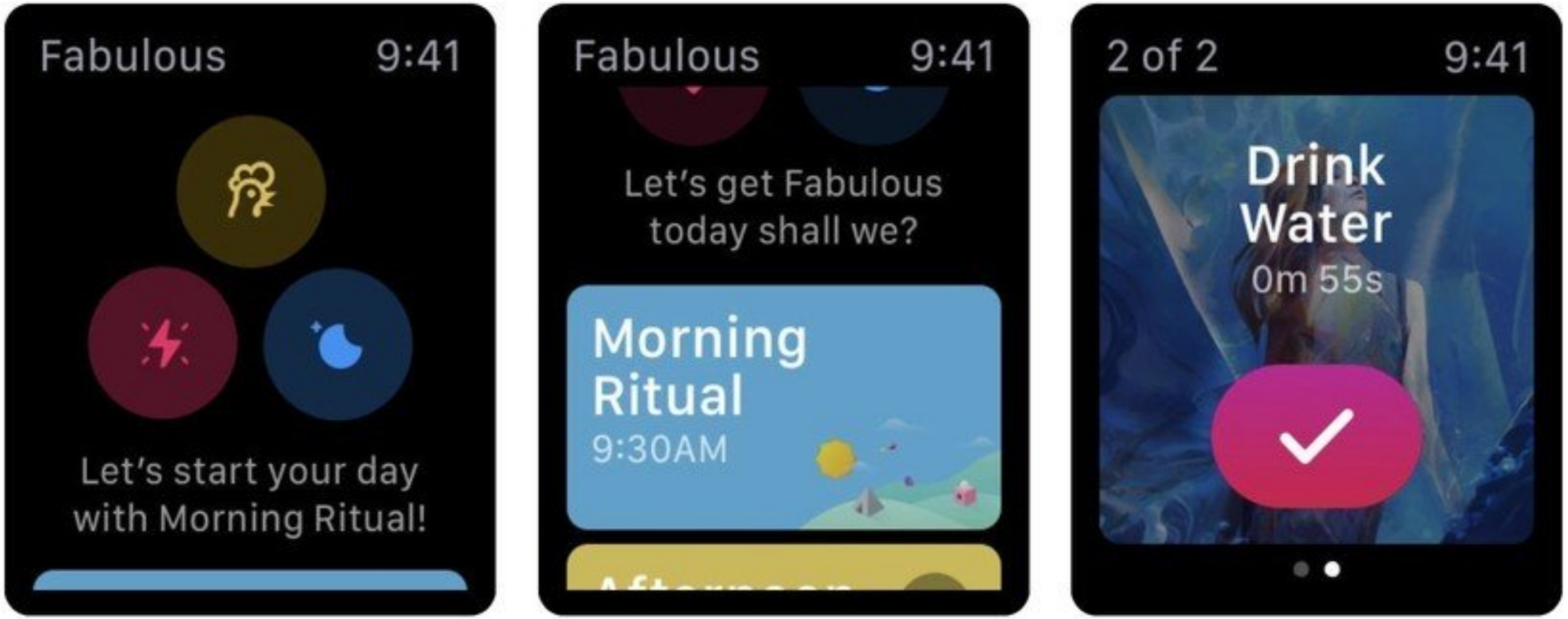
[22] Image source

Experience-Level Patterns

Definition: The *user goals* that make up the *user experience* that the application supports, e.g., activity tracking, coaching, and reviewing.

Experience-level patterns can also capture the *quality* of the user experience, e.g., *motivational* coaching.

Source[23]

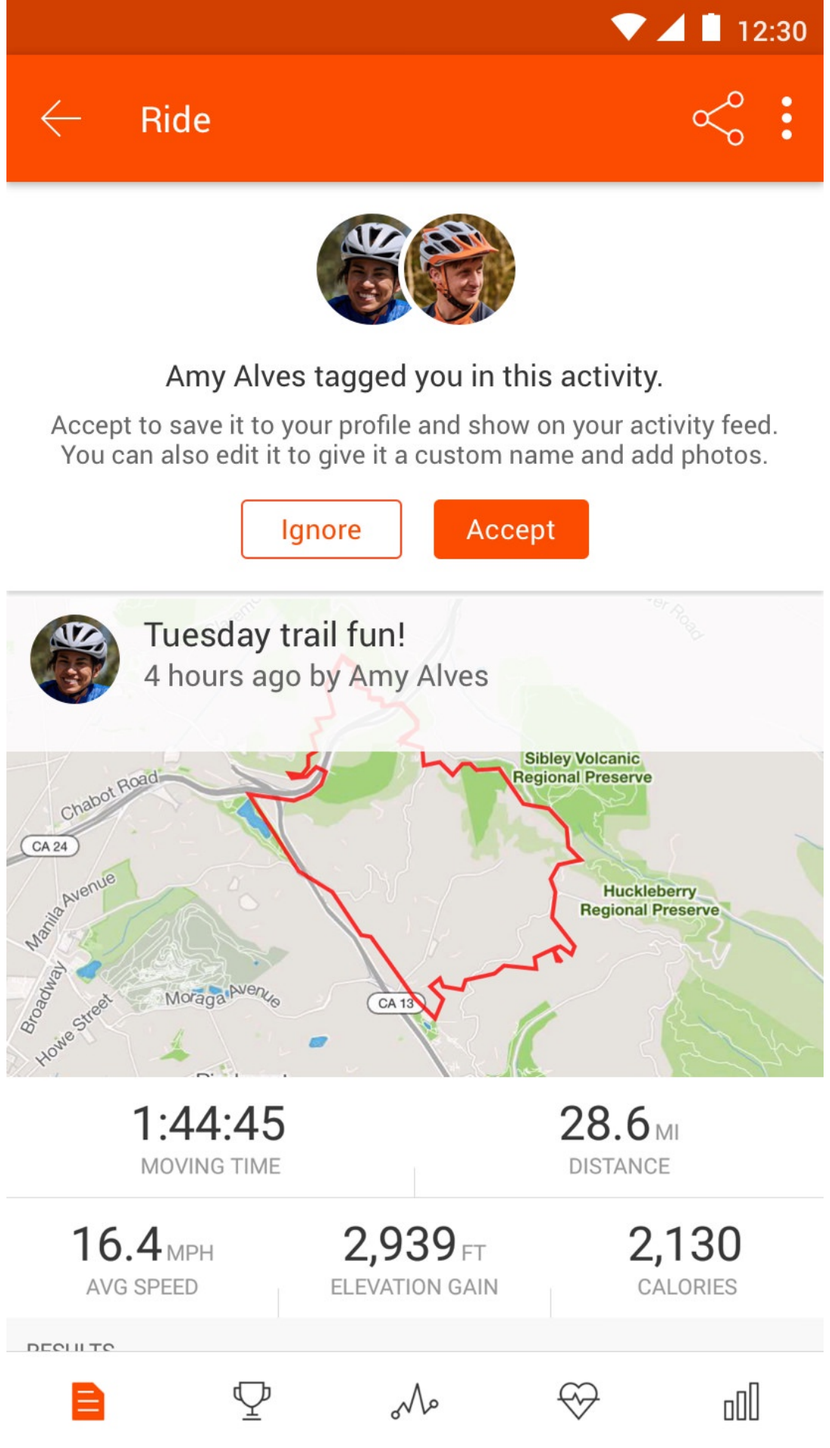


[23] Image source

Elements of an Exprience-Level Pattern[24]

— Primary goals, e.g., activity tracking

— Secondary goals, e.g., community building



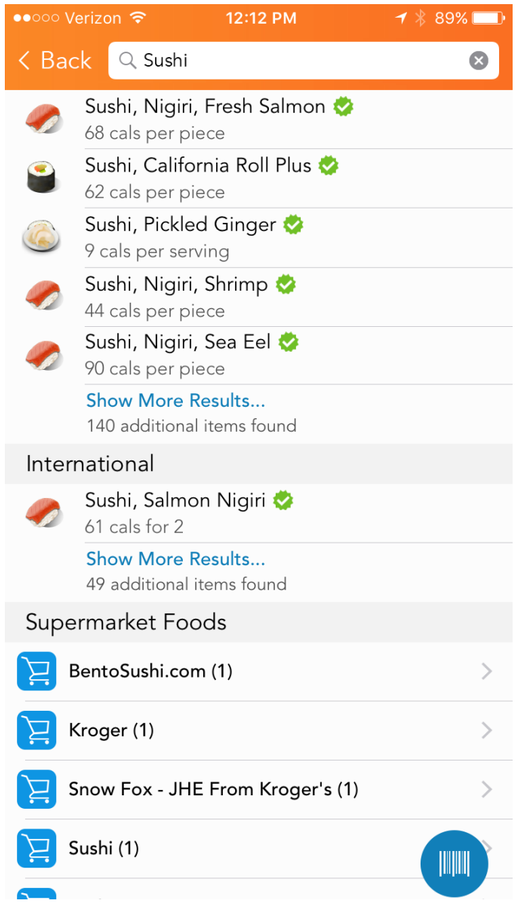
[24] Image source

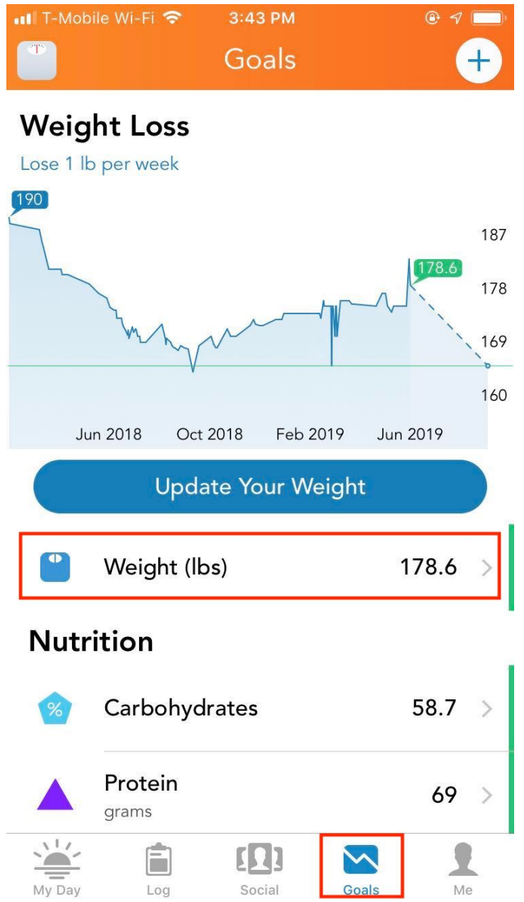
Task-Level Patterns

Definition: Design solutions that help users accomplish sequences of actions that make up user tasks, e.g., logging a meal, capturing a run, or completing a workout.

Tasks point to specific layout patterns. E.g., meal logging can be done through a "search-and-results" pattern, capturing a run can be done through a "dashboard" pattern.

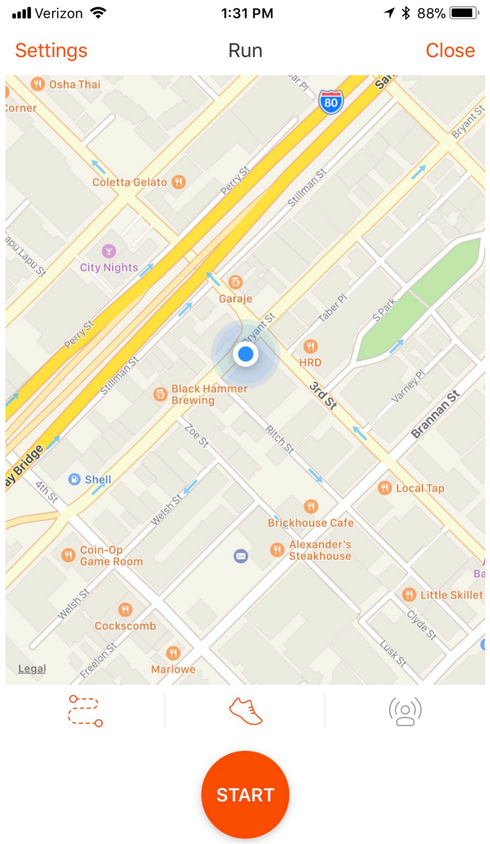
Source[25]

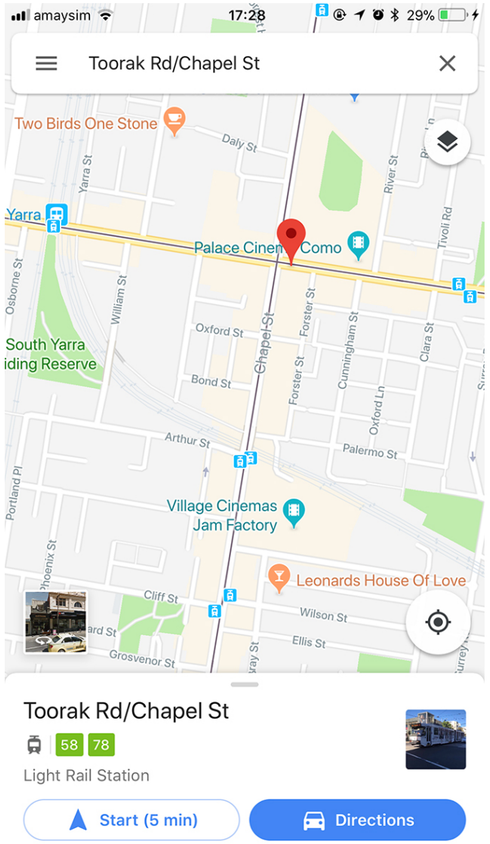




[25] Image sources: left, right

Task-level patterns can be domain independent. Business goals and posture-level patterns set the context for these patterns.[26]





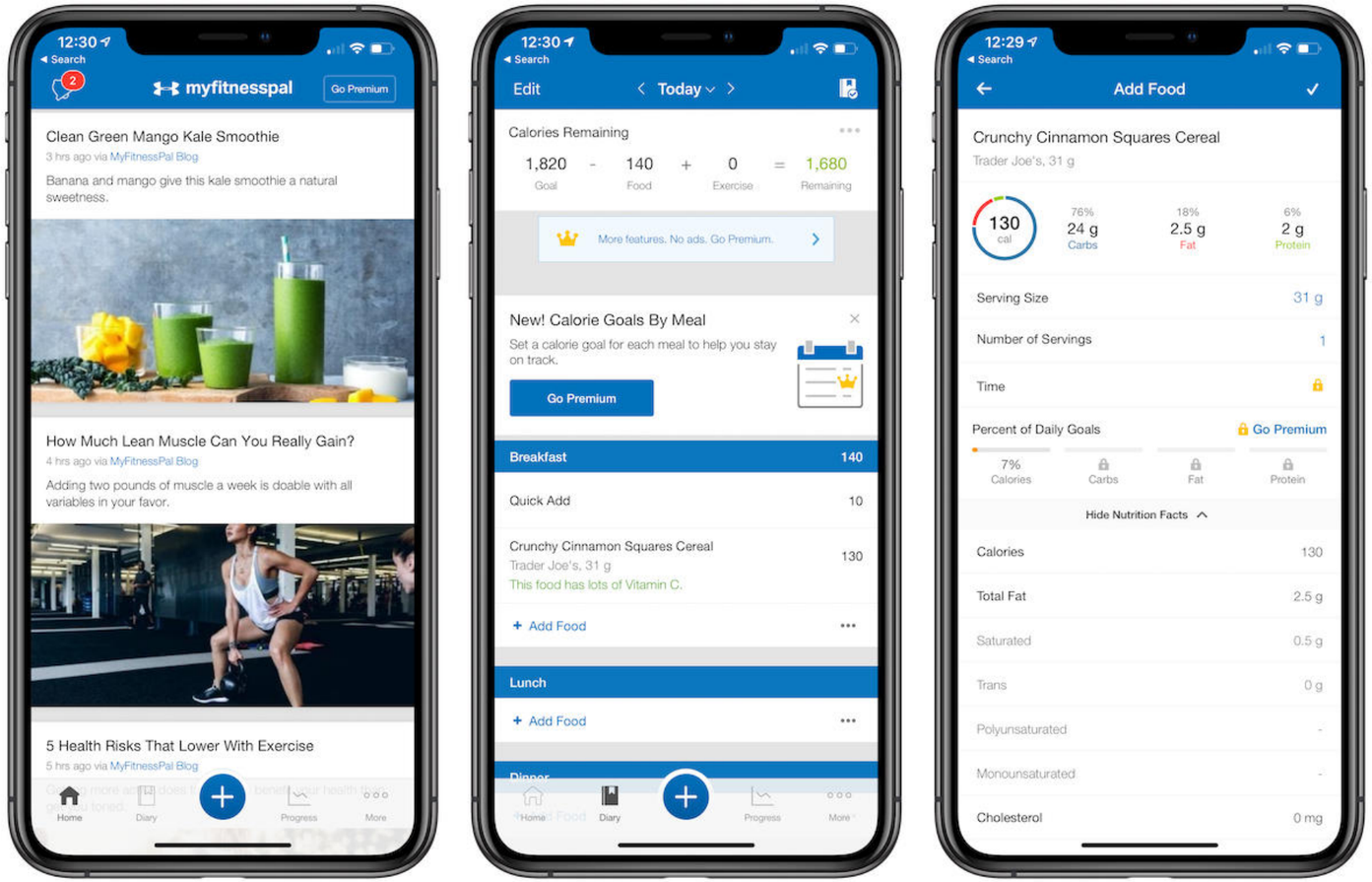
[26] Image sources: left, right

Action-Level Patterns

Definition: Design solutions that support the actions taken to complete the steps(s) of the user's task, e.g., a "start" button to initiate activity tracking, a selectable list entry for a food item.

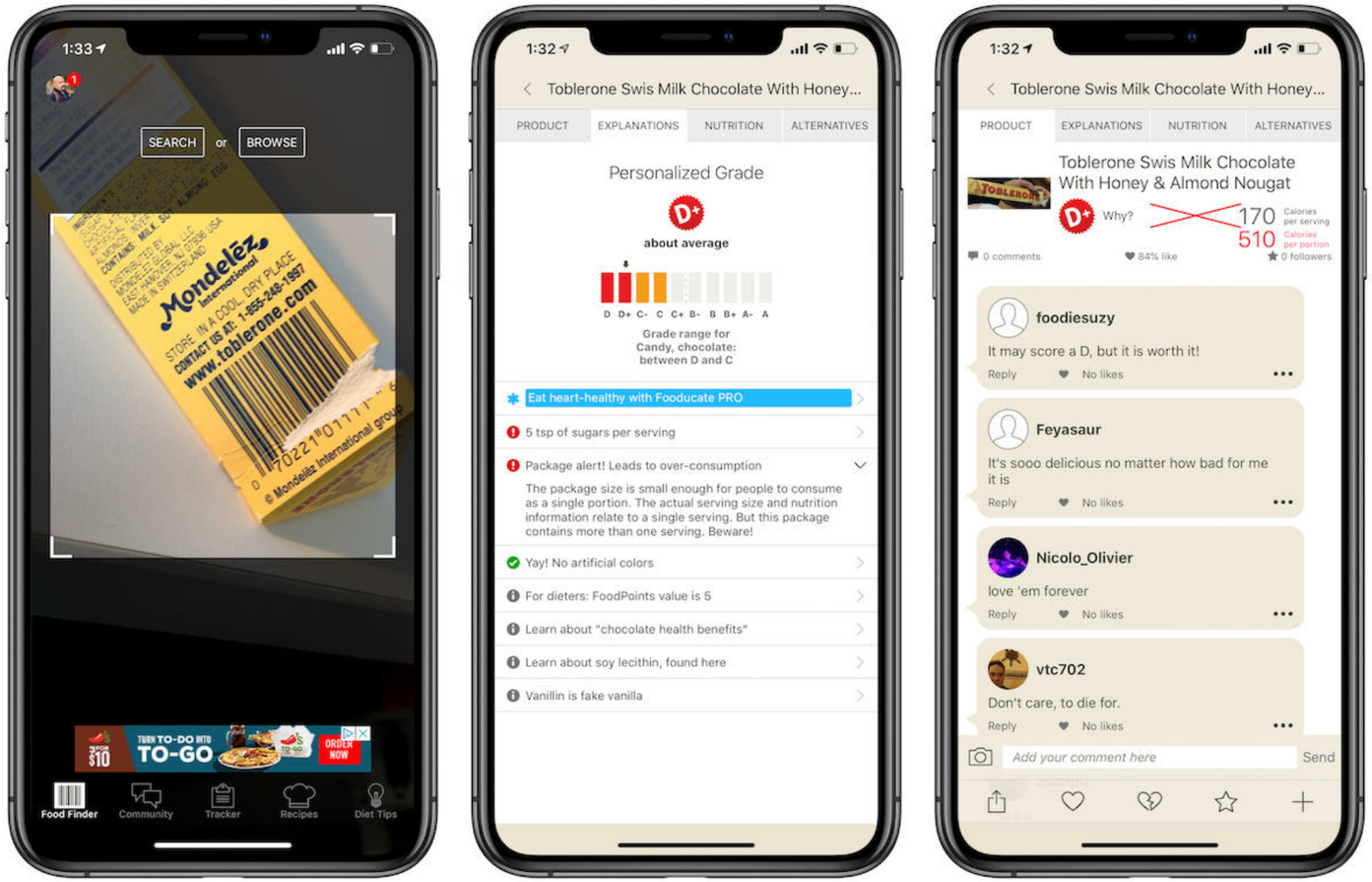
Action-level patterns are the lowest level of building blocks for a design. They are often called *widgets* or *components* (as in React).

Action-level patterns for a *food tracking* app:[27]



[27] Image source: My Fitness Pal

Action-level patterns for a *food education* app:[28]

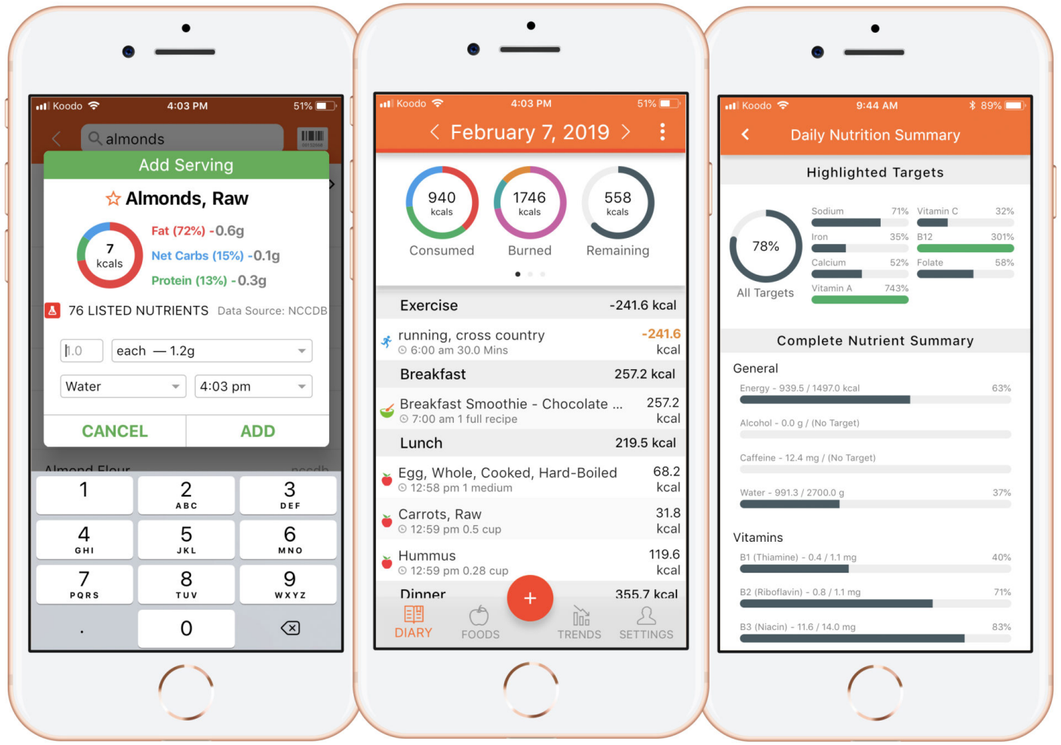


[28] Image source: Fooducate

In-Class Activity

Pattern Language Deconstruction

Source[36]



[36] Image sources: left, right

Business Goals

Mission of the application

Posture Level

“Type” of application

Experience Level

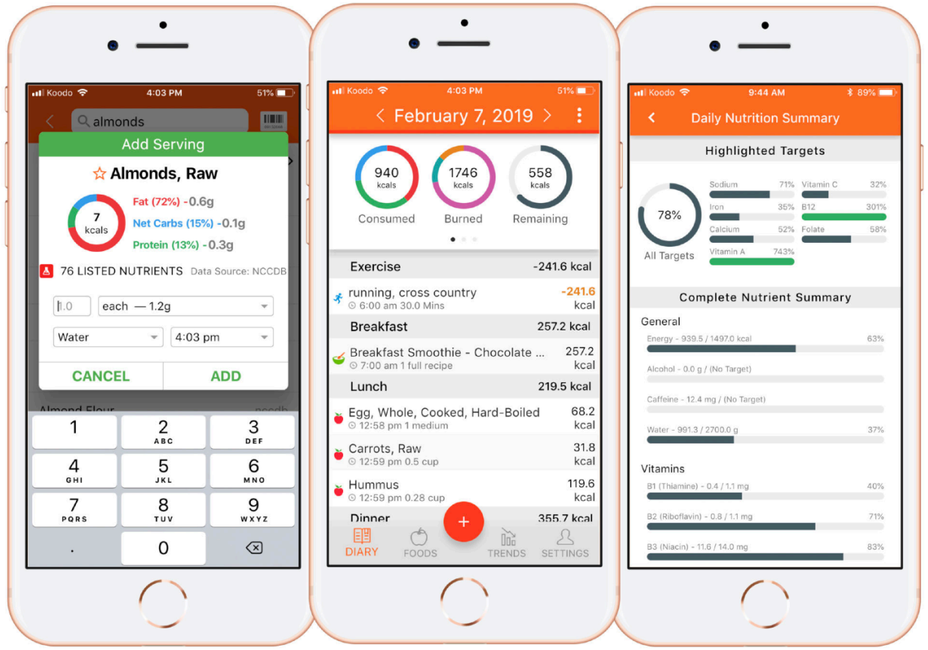
User goals

Task Level

Task sequences

Action Level

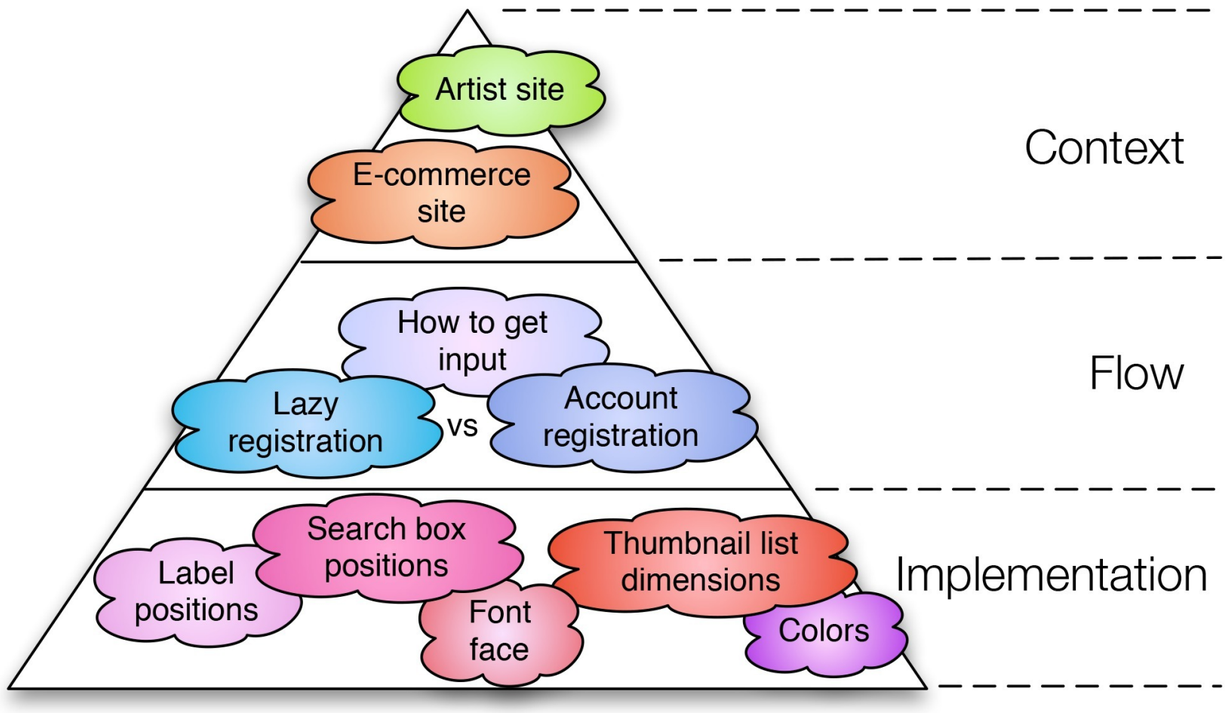
User actions



A Simplifed Model[29] [30]

Three-levels of patterns:

1. **Context:** Type of app
2. **Flow:** Components that support specific functions
3. **Implementation:** The visual/ behavioral elements that implement the functions



[29] Anders Toxboe

[30] More on the three-levels of patterns by Jerry Cao

How do we use patterns?

Common practice: Patterns in the higher levels are defined informally, and the task- and action-level patterns are adopted through experimentation and trial and error.

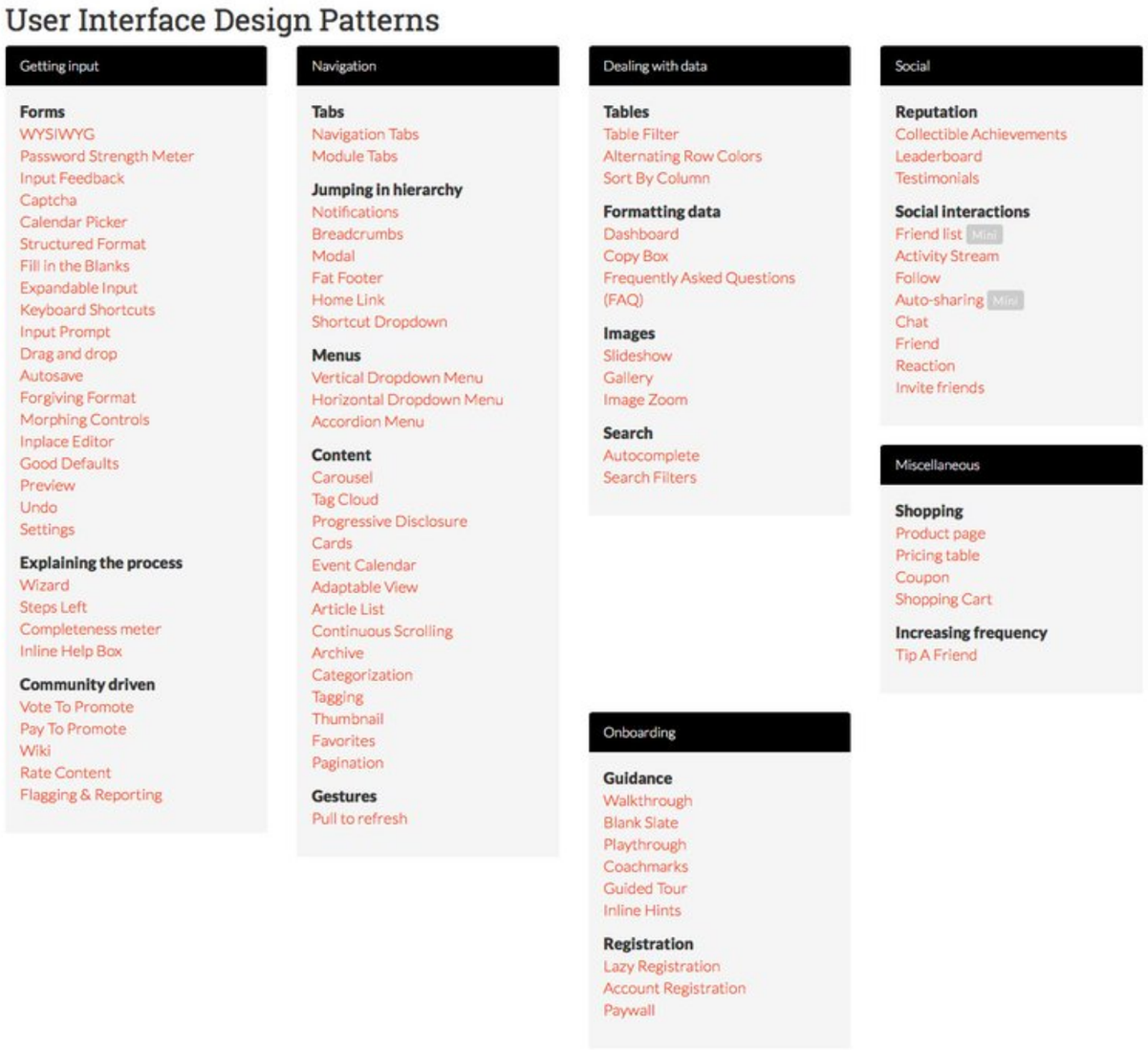
The problem: Ineffective (e.g., lack of coherence across different levels) and inefficient (wasted effort in experimentation).

The solution: Defining patterns top to bottom will "generate" the design when patterns are available across all levels.[31]

[31] van Welie & van der Veer, 2003

Where do we find patterns?[32]

Task- and action-level patterns are organized into catalogues/collections based on functional similarity.



[32] Image source

Online Pattern Libraries

- UIPatterns.io
- UI-Patterns
- Mobbin
- UI Garage
- Welie

Design Style Guides

Definition: A vocabulary of design elements that are repeatedly applied to interaction design problems. These are task- and action-level interface components that follow a consistent look and feel in appearance and behavior.

Non-digital example: NASA Graphics Standard Manual.[33]

[33] NASA

NASA Uniform Patches

Personnel identification is an important facet of the NASA identification program. An embroidered patch incorporating the logotype is available for application on a wide variety of uniforms and clothing. Two patch designs, shown to the right, are available.

For general personnel, a white patch with a NASA Red logotype is available. This achieves the simplest and most effective identification on various types and colors of clothing that may include other badges or name tags. The patch is applied on the right front side of the garment approximately 1½" (3.8 cm) directly above the breast pocket or in a comparable position on garments without pockets. On a blazer (fig. e), the top edge of the patch aligns with the left breast pocket.

A few specific color recommendations are made for NASA uniforms: royal blue for flight suits; white for lab coats, hardhats, and helmets. A 7" wide (17.8 cm) logotype may be embroidered in NASA Red centered on the back of a white lab coat (fig. d). On a white hardhat or helmet, a 5" wide (12.7 cm) NASA Red decal of the logotype may be centered on the front (fig. g).

To distinguish emergency/security personnel (security guards, firemen, etc.) a distinctive NASA Red patch with a white border, white logotype and the installation identification in black is available. The name of the emergency/security service (i.e. Fire Department) appears in white centered within a smaller black patch that is positioned ⅜" (.9 cm) under the red patch. This configuration is worn on both shoulders of the uniform, on both shirts (fig. f) and outer-jackets. A light blue shirt and hat with dark blue trousers or skirt is recommended.

General personnel patch

Emergency/security patches

a) Flight jacket

b) Shirt/blouse

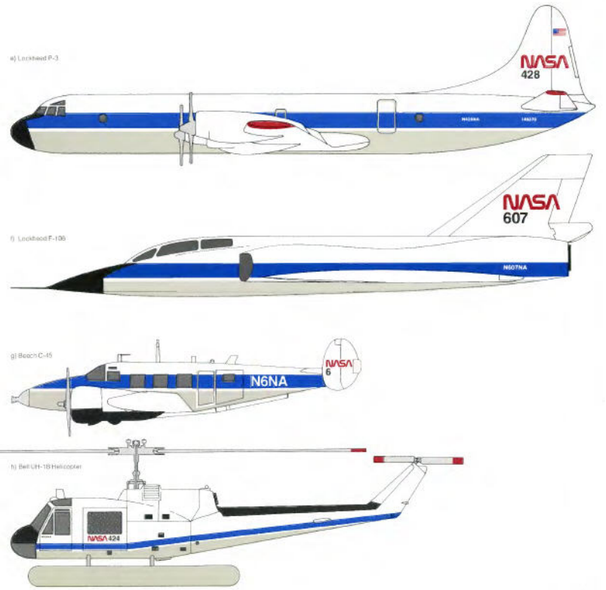
c) Flight suit/mechanic suit

d) Laboratory coat

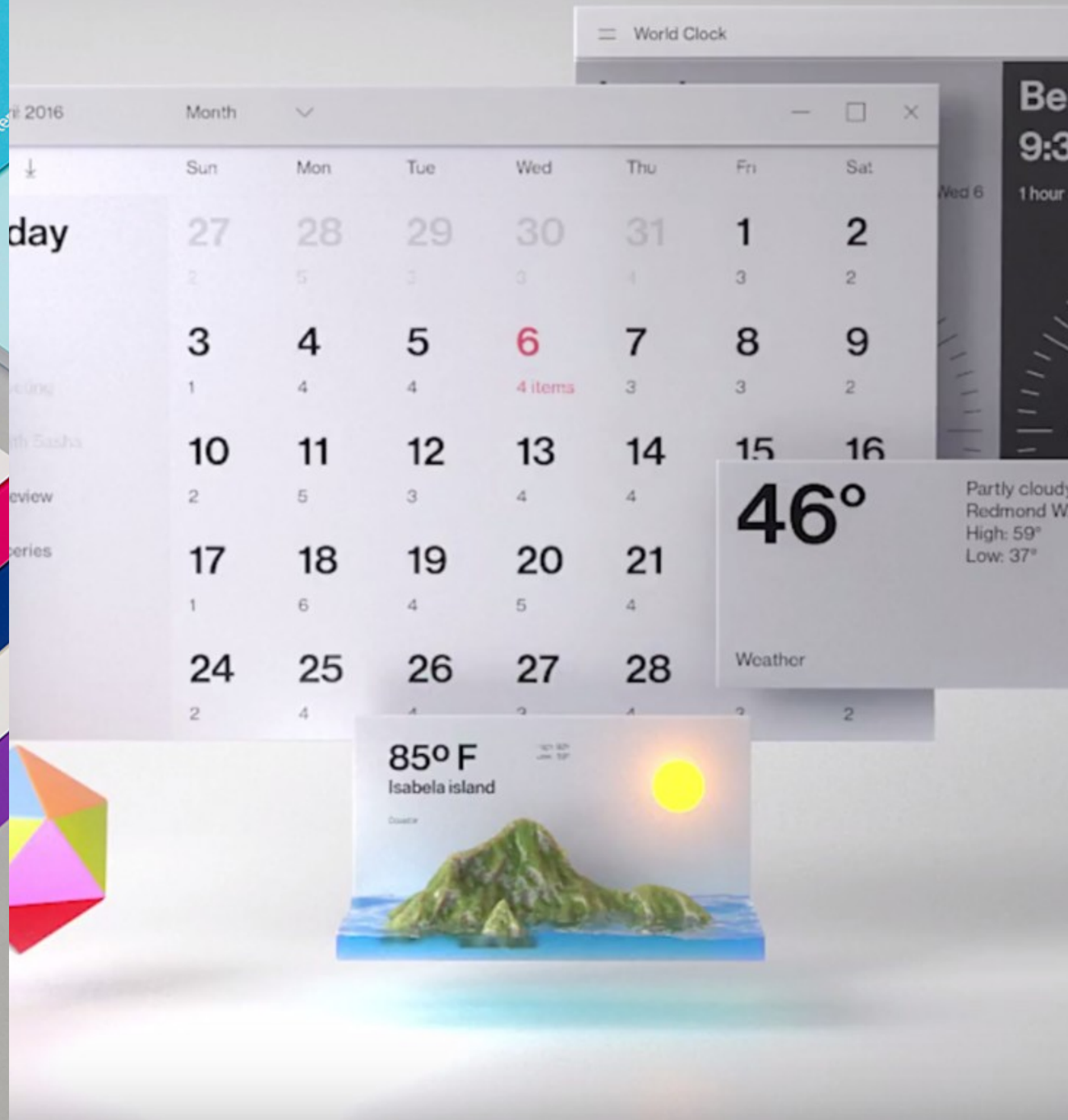
e) Blazer/sport jacket

f) Emergency/security shirt (side view)

g) Hardhat/helmet

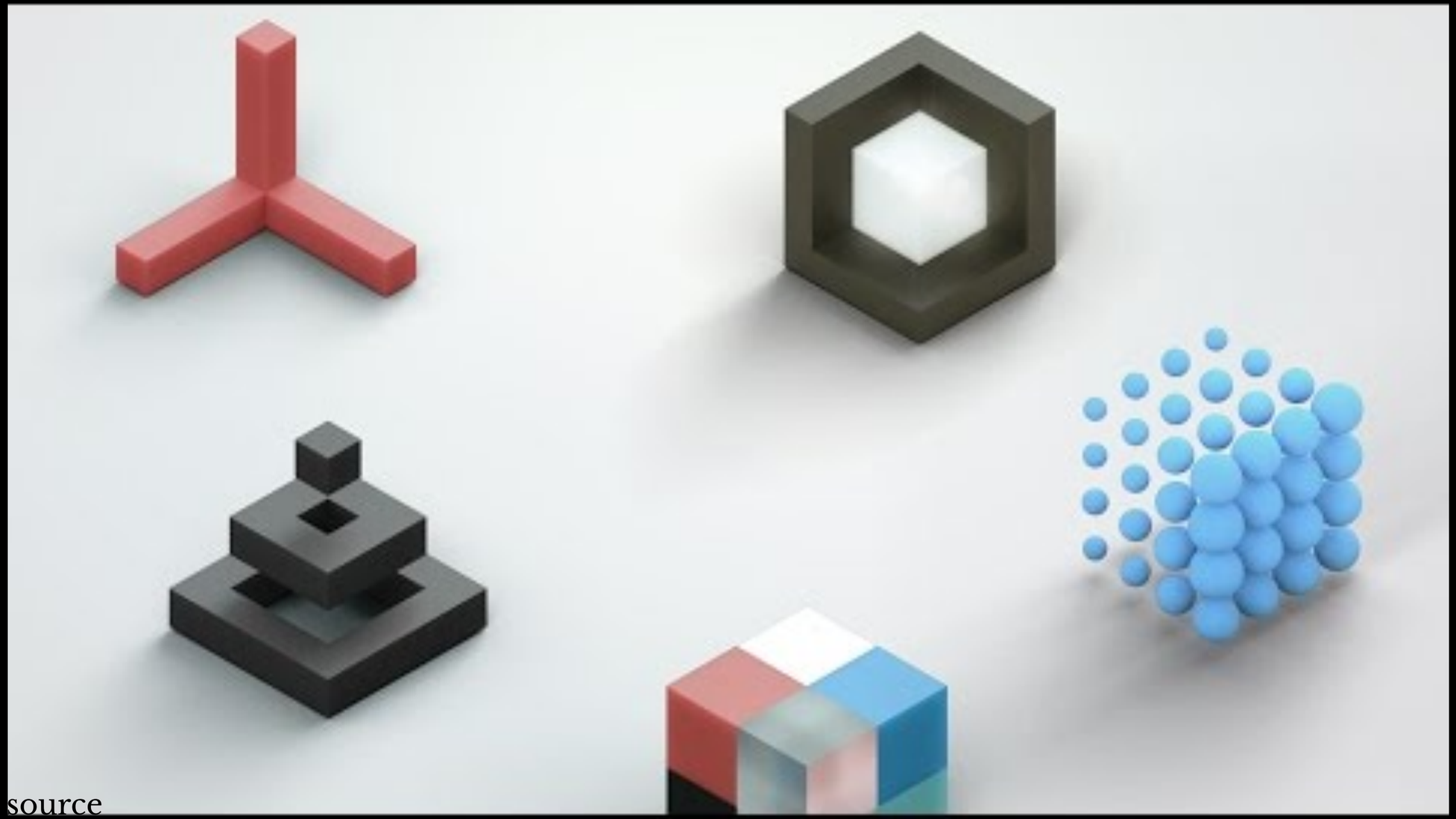


Source[3435]



[34] *Left:* Google Material Design

[35] *Right:* Microsoft Fluent Design System



source

Commonly Used Design Style Guides[20]

- Material Design
- Fluent Design System
- Materialize
- Ant Design
- Grommet
- Flat Remix

[20] Image source

Case Studies of Design Language Use

— Material studies examples

— Fluent design case studies

What did we learn today?

— Design paradigms
— Design patterns
— Design languages