SWE 632 - Design & Development of User Interfaces

Spring 2021

George Mason University

Dr. Kevin Moran

Week 10:

Preventing Error

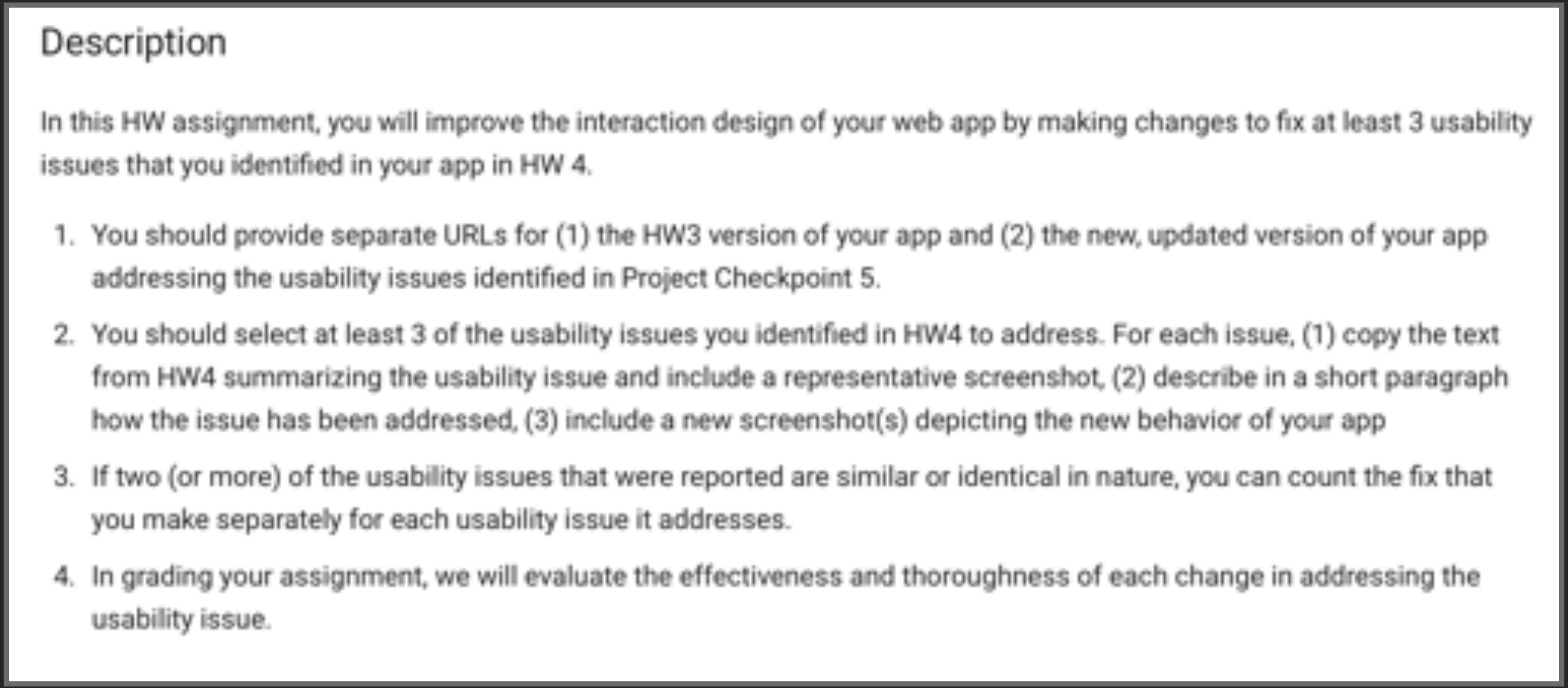
Administrivia

- Project Checkpoint 5 ***due next week***, April 7th
- Project Checkpoint 6 ***due in two weeks***, April 14th
- Discussion Question 10 - Posted after class

Project Checkpoint 5 - Defining Tasks

- ***Suggestions for Usability Tasks:***
  - Should strike a balance, not too specific, not too general
  - Should have a clear goal for the user (e.g. “purchase an item from the online store”, not “shop in the online store”
  - Can include a “preamble” or background information.
    - For example, if you need to educate a user, or set the scene.

Project Checkpoint 6: Interaction Design #2



Description

In this HW assignment, you will improve the interaction design of your web app by making changes to fix at least 3 usability issues that you identified in your app in HW 4.

1. You should provide separate URLs for (1) the HW3 version of your app and (2) the new, updated version of your app addressing the usability issues identified in Project Checkpoint 5.
2. You should select at least 3 of the usability issues you identified in HW4 to address. For each issue, (1) copy the text from HW4 summarizing the usability issue and include a representative screenshot, (2) describe in a short paragraph how the issue has been addressed, (3) include a new screenshot(s) depicting the new behavior of your app
3. If two (or more) of the usability issues that were reported are similar or identical in nature, you can count the fix that you make separately for each usability issue it addresses.
4. In grading your assignment, we will evaluate the effectiveness and thoroughness of each change in addressing the usability issue.

Class Overview

1. *Human Error:* Understanding why Humans Make Mistakes
2. *Designing for Error:* Designing to Help Prevent Error
3. *Direct Manipulation:* Acting “Physically” upon objects
4. *7 Minute Break*
5. *Group Activity:* Designing a Direct Manipulation App
6. *Tech Talk:* Django
7. *Tech Talk:* NodeJS

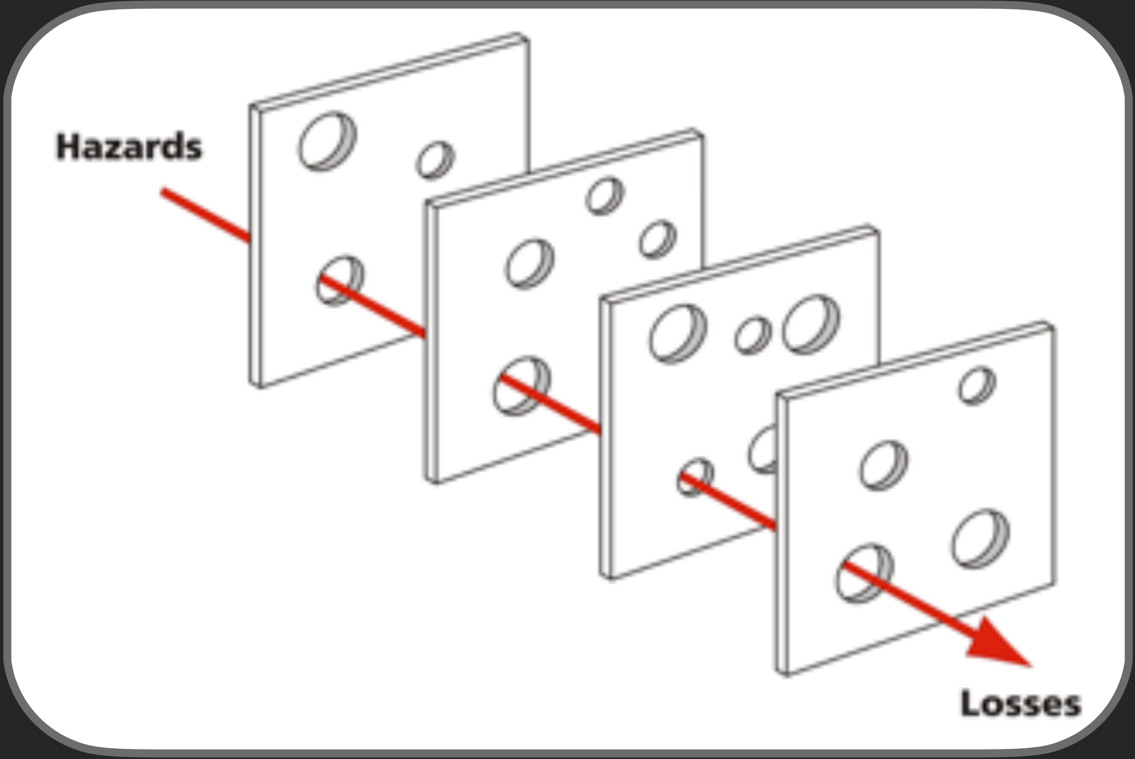
Human Error

What Causes Disasters?

- Mechanical malfunction?
- Poor design?
- Human error?

Swiss Cheese Model

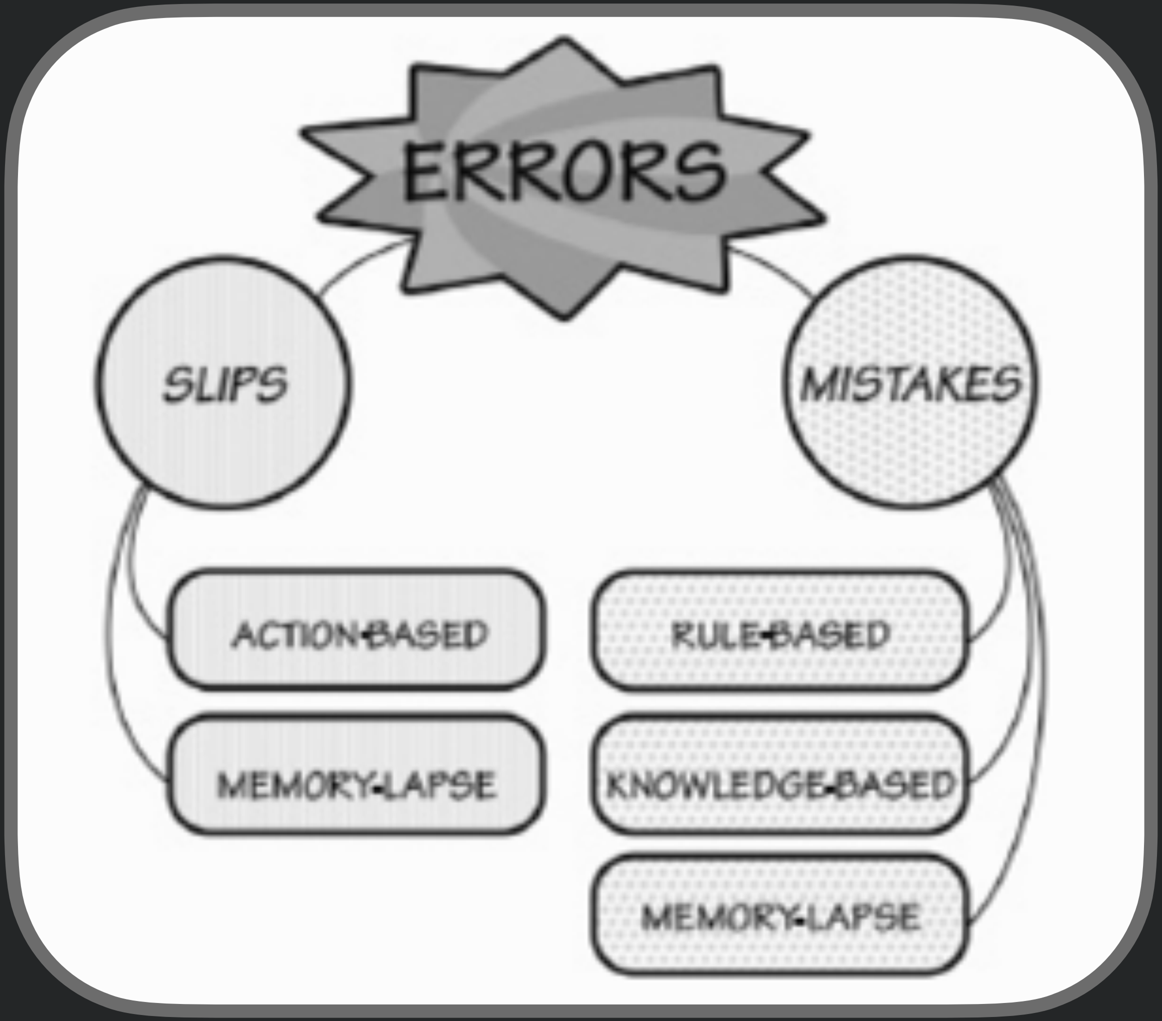
- Accidents must penetrate levels of system defenses
- Reduce accidents by
  - Adding more layers
  - Reduce the size and number of holes
  - Alert users when holes line up



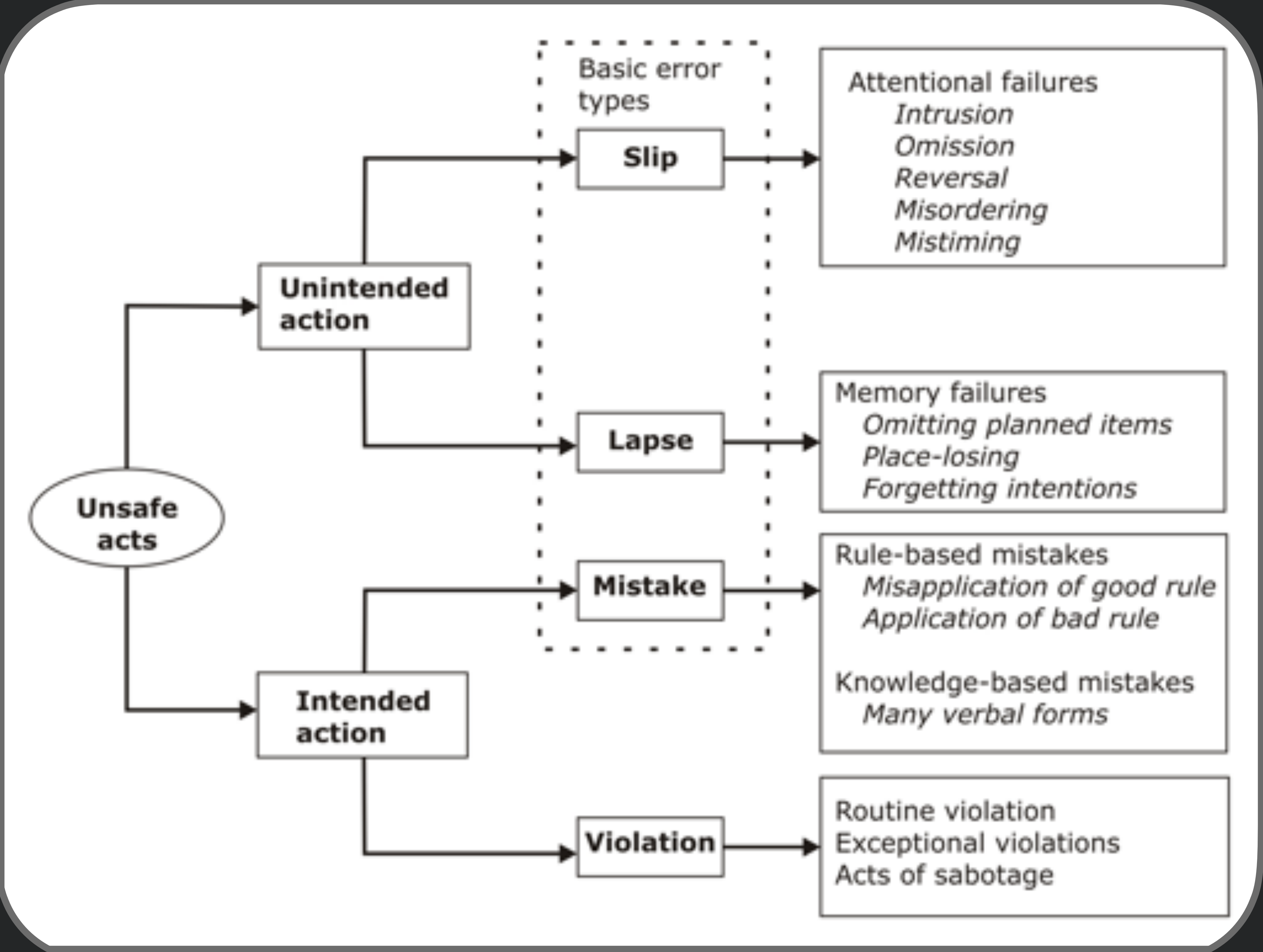
Root Cause Analysis

- Keep asking ***why*** to determine causes for erroneous actions, and the causes of these causes
- Example
  - 2010 F-22 crash that killed pilot
  - Official cause: pilot error - pilot failed to take corrective action
  - Why did the pilot not take the action?
    - Pilot was not receiving oxygen and was probably unconscious.

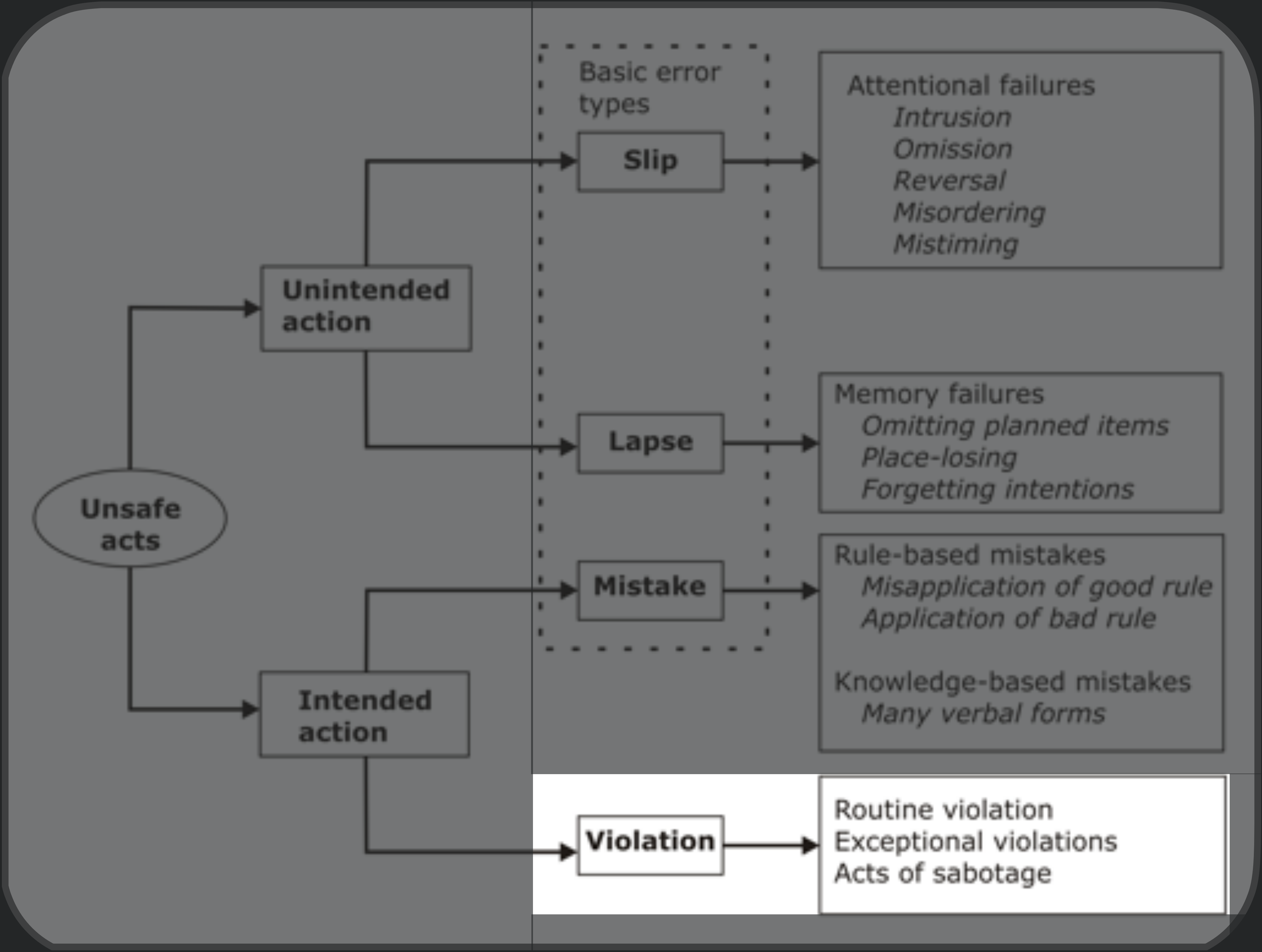
Psychological Types of Unsafe Acts



Psychological Types of Unsafe Acts



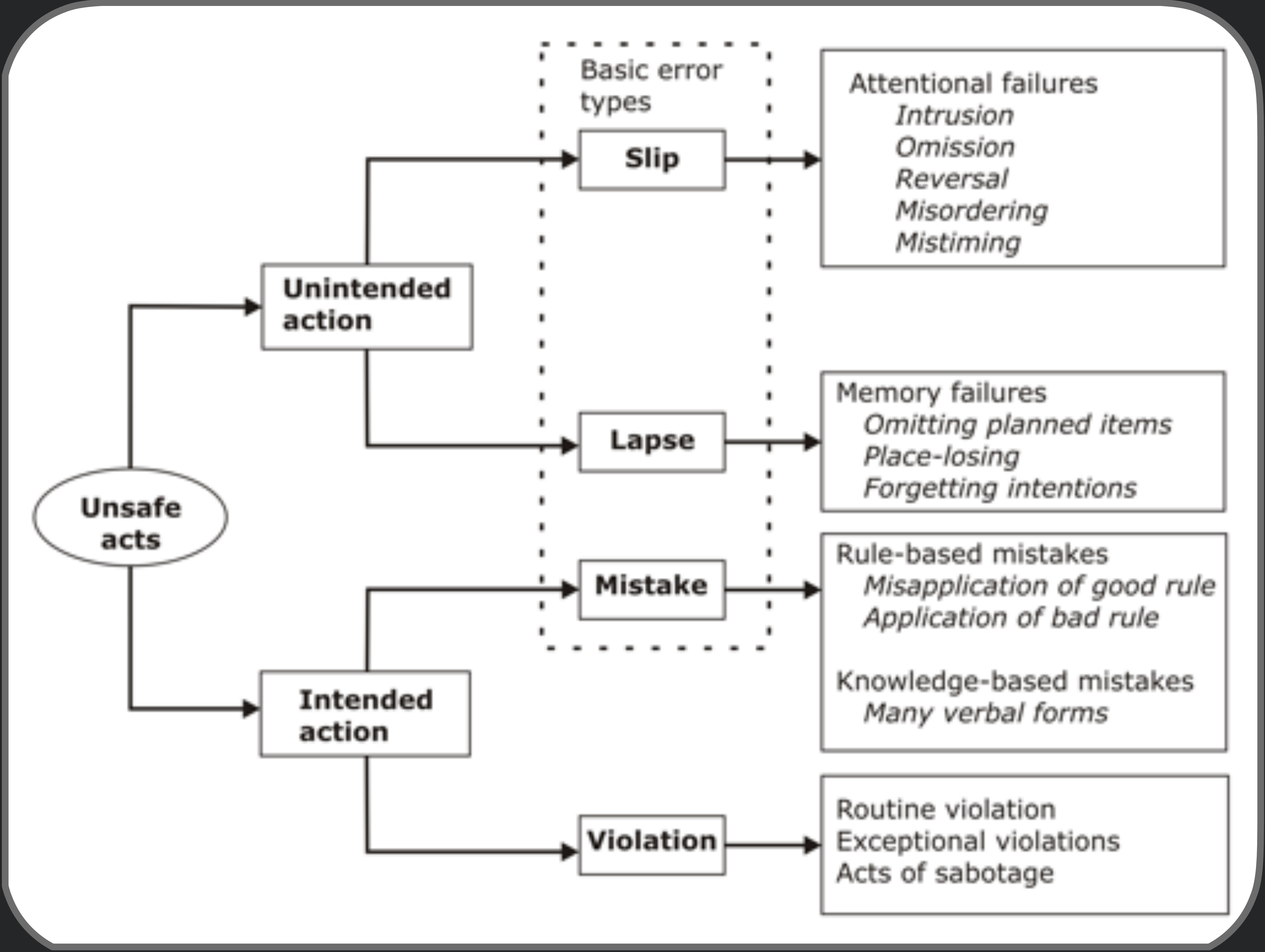
Psychological Types of Unsafe Acts



Deliberate Violations

- Error occurred because user ***intended*** the erroneous output
- Routine violation - user always intends to do it
  - Noncompliance is so frequent it is ignored
  - E.g., running a red light
- Exceptional - only in some cases
- Sabotage - intended destruction

Psychological Types of Unsafe Acts



Basic error types

Unsafe acts

Unintended action

Intended action

Slip

Lapse

Mistake

Violation

Attentional failures
- *Intrusion*
- *Omission*
- *Reversal*
- *Misordering*
- *Mistiming*

Memory failures
- *Omitting planned items*
- *Place-losing*
- *Forgetting intentions*

Rule-based mistakes
- *Misapplication of good rule*
- *Application of bad rule*

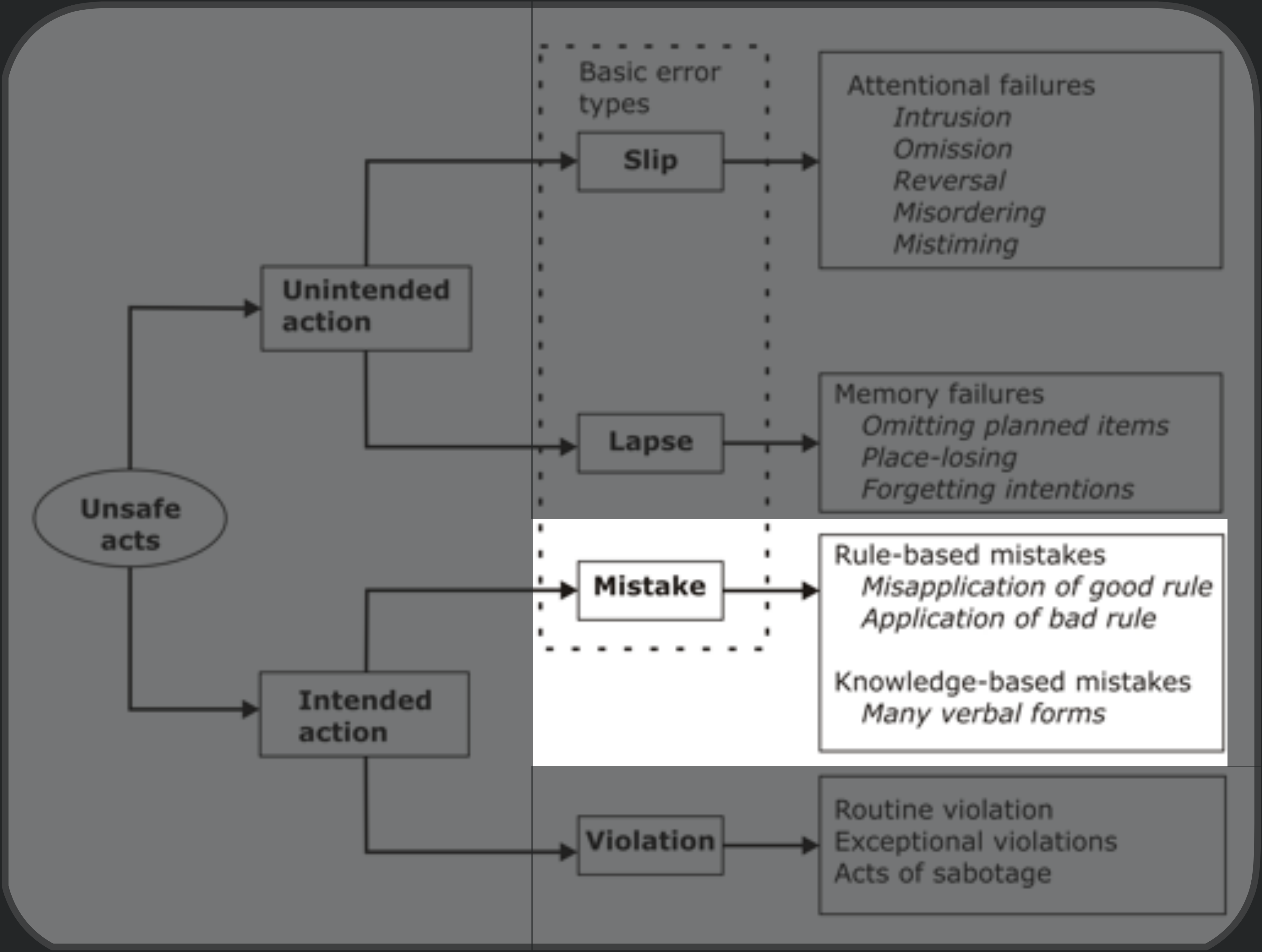
Knowledge-based mistakes
- *Many verbal forms*

Routine violation

Exceptional violations

Acts of sabotage

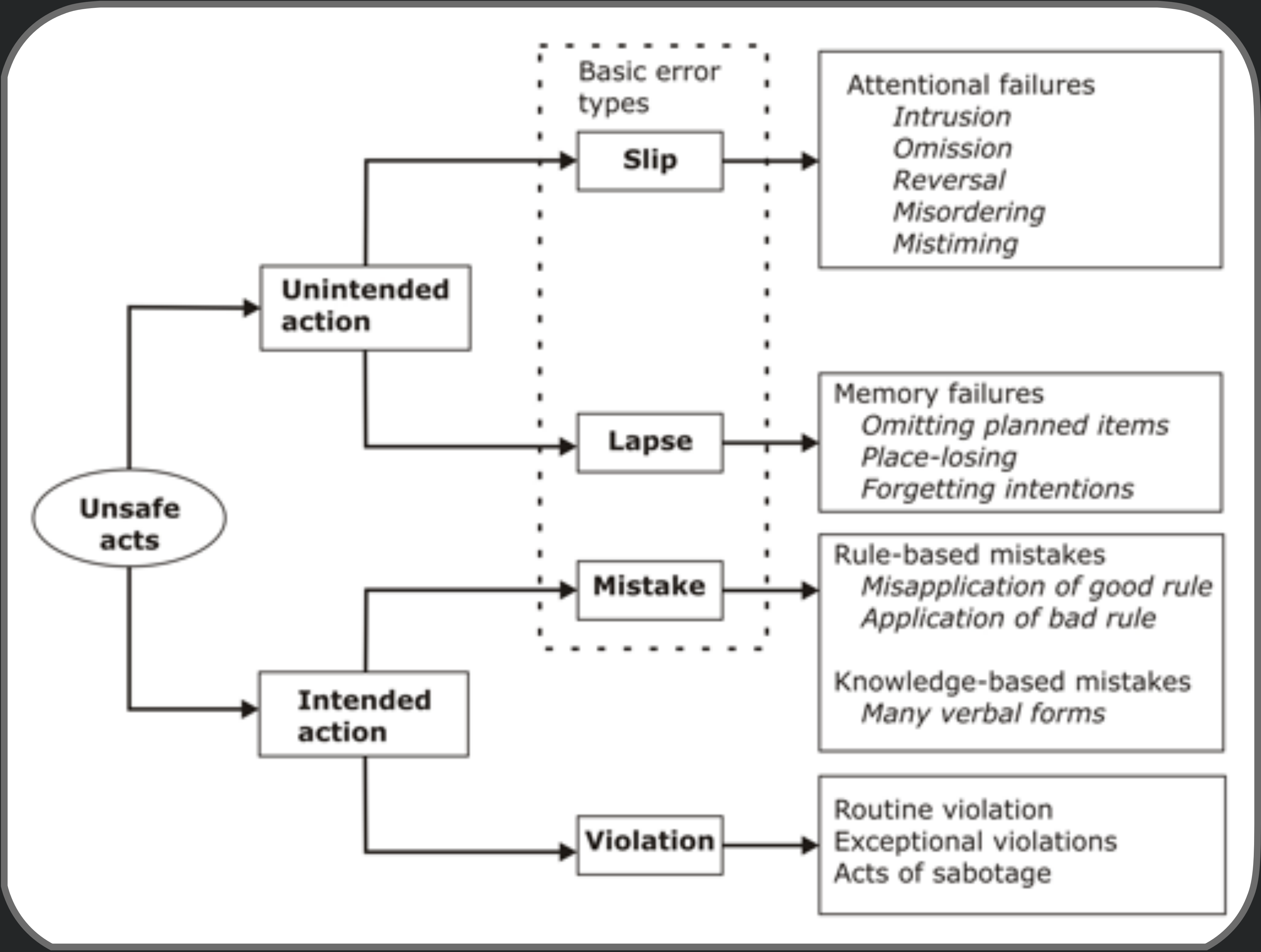
Psychological Types of Unsafe Acts



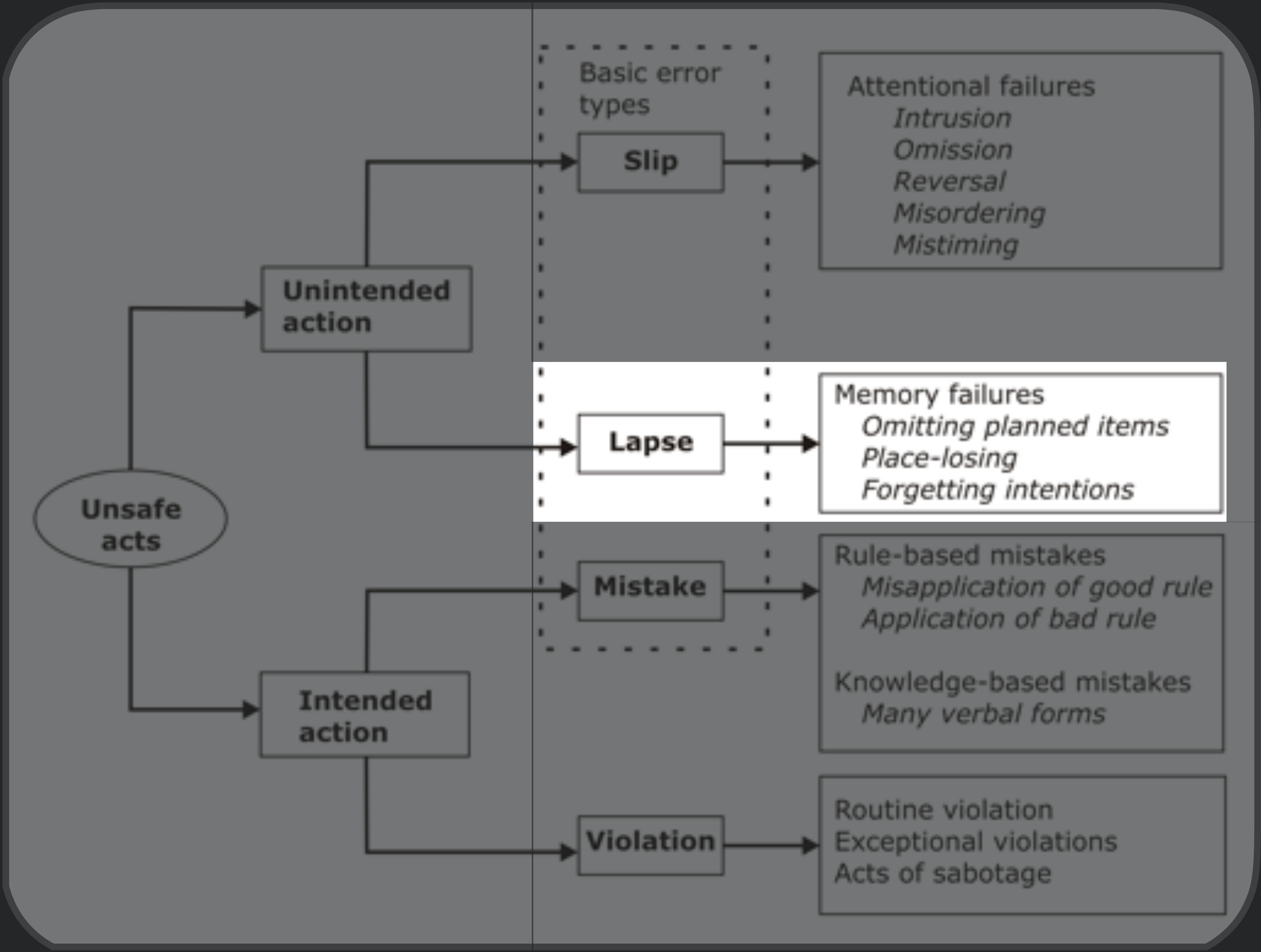
Mistakes

- User ***formulated*** the wrong goal or plan
    - Executing action will not achieve goal
- Rule based: appropriately diagnosed situation, but chose erroneous course of action
    - Example: Night club attendees blocked from leaving during fire because bouncers thought they were breaking rules
- Knowledge based: does not have correct information
    - Example: Skidding driver feels brake vibrations, believes indicates malfunctioning breaks and takes foot off break, stopping ABS

Psychological Types of Unsafe Acts



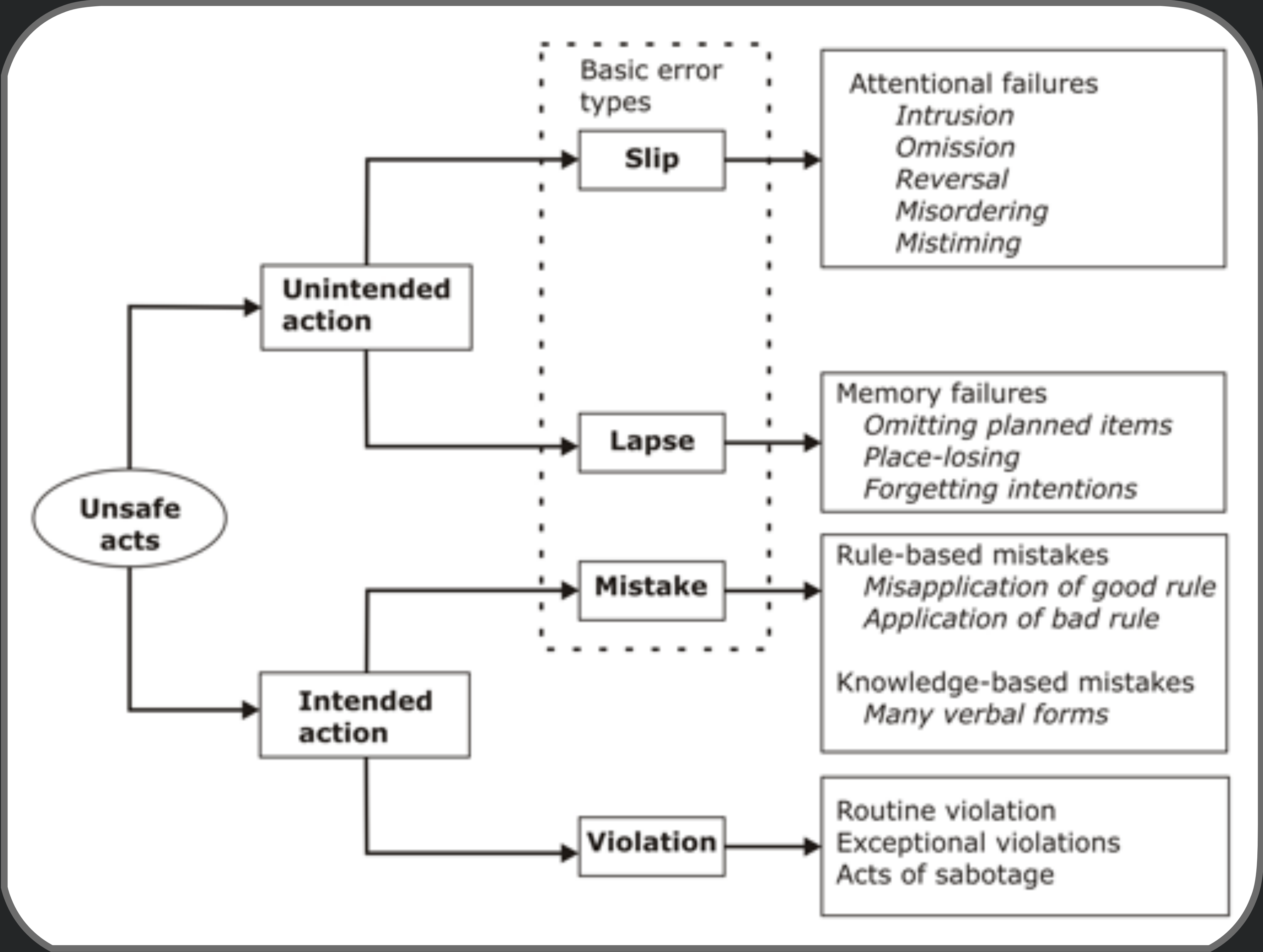
Psychological Types of Unsafe Acts



Memory Lapse

- Failing to do all steps of a procedure, repeating steps, forgetting the outcome of an action, forgetting the goal or plan
- Often caused by interruption
  - Time between when plan was formulated and plan was executed leads to forgetting plan
  - Take a pen out to sign form, get interrupted talking to someone, leave it on desk rather than put it back in bag

Psychological Types of Unsafe Acts



Unsafe acts

Unintended action

Intended action

Basic error types

Slip

Lapse

Mistake

Violation

Attentional failures
Intrusion
Omission
Reversal
Misordering
Mistiming

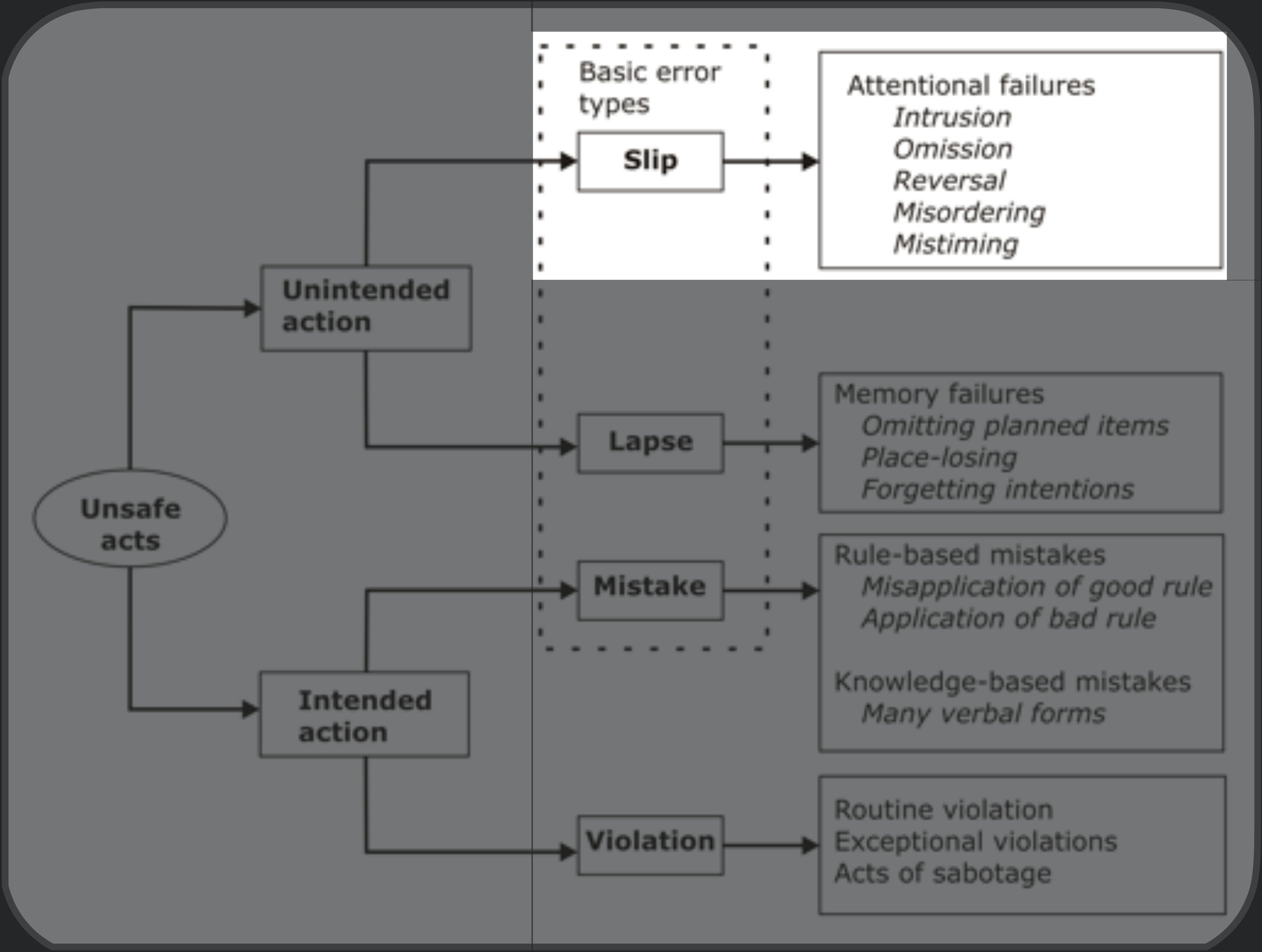
Memory failures
Omitting planned items
Place-losing
Forgetting intentions

Rule-based mistakes
Misapplication of good rule
Application of bad rule

Knowledge-based mistakes
Many verbal forms

Routine violation
Exceptional violations
Acts of sabotage

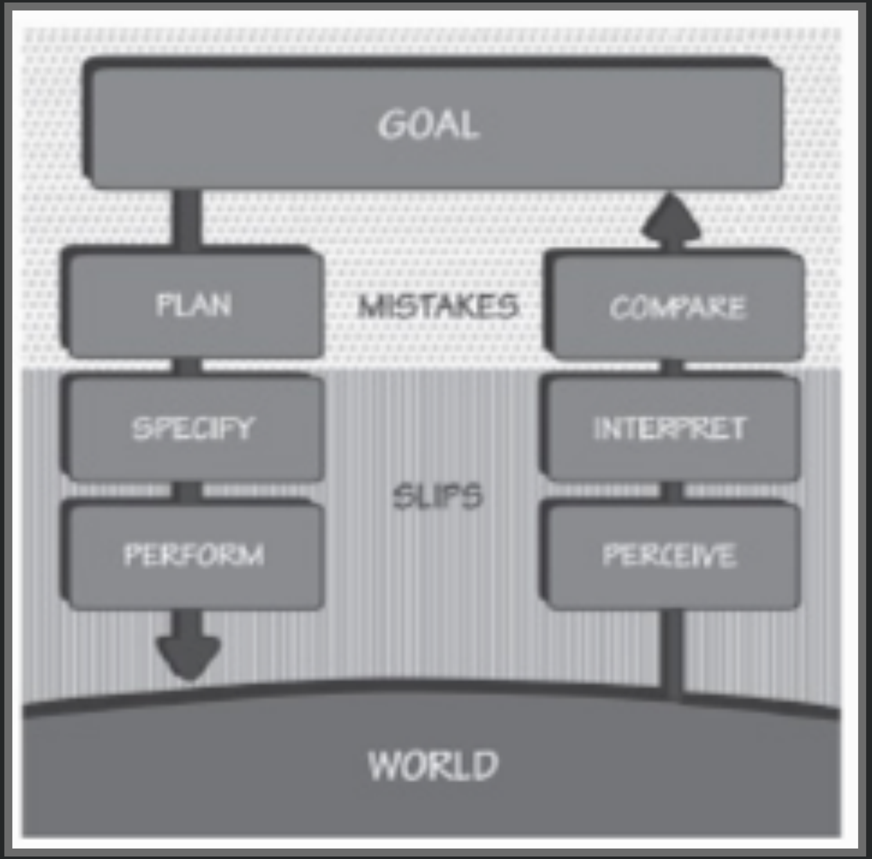
Psychological Types of Unsafe Acts



Slips

- Attentional failure - user ***intended*** to do correct action, but did not actually execute action
- Example: I poured some milk into my coffee and then put the coffee cup into the refrigerator. This is the correct action applied to the wrong object.

Error & the Seven Stages of Action



- ***<u>Novices</u>*** *are more likely to make mistakes than slips, and* ***<u>experts</u>*** *are more likely to make slips.*

Potential Underlying Causes

- Strong Habit Intrusion
- Omissions
- Perceptual Confusion
- Mistimed Checks

Strong Habit Intrusion

- Performance of some well-practiced activity in familiar surroundings
- Intention to depart from custom
- Failure to make an appropriate check
- Example: start trip to frequent destination, forget going somewhere else

Omissions

- May be interrupted, forgetting intention to act
- “I picked up my coat to go out when the phone rang. I answered it and then went out of the front door without my coat.”

Perceptual Confusion

- Take frequent action very often, leading to high System 1 automation
- Don't perform perceptual check to verify that System 1 action is the correct one to take
- Example: "I began to pour coffee into the sugar bowl"

Mistimed Checks

- Highly automated System 1 activity that is interrupted
- Error in resuming activity because usually unconscious.
- Example - interrupted in the middle of tying shoes

Mistimed Checks

- Highly automated System 1 activity that is interrupted
- Error in resuming activity because usually unconscious.
- Example - interrupted in the middle of tying shoes

Activity: Raise your Hand

Activity: Raise your Hand

- Think of the last unsafe act you performed in a piece of software.

Activity: Raise your Hand

- Think of the last unsafe act you performed in a piece of software.

Activity: Raise your Hand

- Think of the last unsafe act you performed in a piece of software.

- What was the underlying cause?

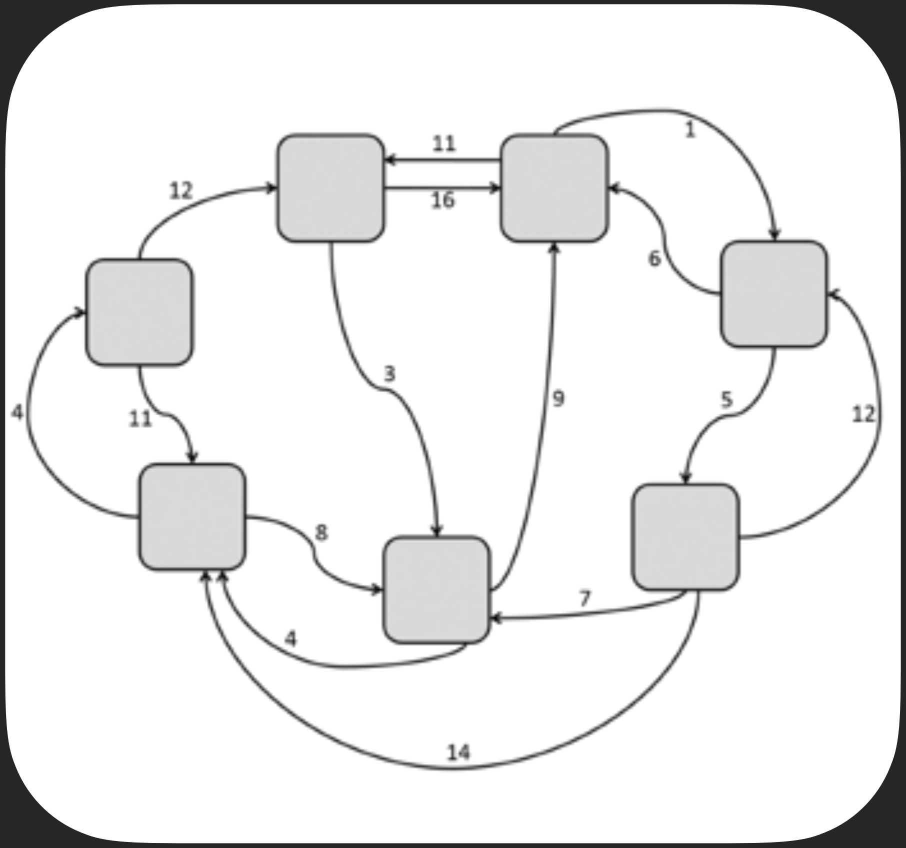
Designing for Error

Designing for Error

- Humans are not automatons and will never behave like automatons
- Easy to design for the situation in which everything goes well
- But important to think about what might go wrong and how the interaction design can ameliorate issues

Information Foraging Theory Perspective

- Information Foraging Theory (IFT) perspective
  - User exploring patches topology in search of prey
  - Always making a decision about whether a patch is the right place to hunt and changing as new information arrives



- Breaks down when user actions transform the state of the application
  - Patches and topology no longer fixed
  - Visiting a configuration of the system by clicking "Send" on the email editor is a not an undoable action

Some Strategies for Designing for Errors

- Understand the cause, and fix it
- Make it possible to reverse errors
- Offer feedback that enables users to discover and correct errors
- Don't treat actions as errors, but as manipulations

Understand the Causes of Errors

- What errors occur? What type are they? How can they be prevented?
- Frequent contributing factors
  - Ambiguous or unclear information about the state of the system
  - Lack of an effective conceptual model
  - Inappropriate procedures
- Must design for users as they exist, rather than users as you'd like them to behave

Interruptions

- Interruptions are a frequent cause of error
- User may be using your interface perfectly, with the correct plan to get to their goal
    - What happens if, in the middle of the task, they answer a phone call?
    - Or if they run out of time, and come back the next day?

Designing for Interruptions

- Help user resume task, by remembering where they were in task, what steps have been completed, and what steps remain

- Reduce the number of steps

- Use forcing functions to force users to do forgettable action (e.g., take card from ***before*** picking up cash)

Brief Activity: Interruptions

- In your project groups

- Imagine a user was interrupted while using one of your project apps
- What errors might this create?
- What challenges might users experience when resuming?

- How could you change your design to address these issues?

Offer Feedback for User Actions

- Feedback helps keep users on track in accomplishing goals
  - Provide feedback early
  - Provide feedback consistently
- Make feedback visible, noticeable, legible, located w/ in users focus of attention
- Requesting confirmation can be used to prevent costly errors (but use sparingly)

Tone of Feedback

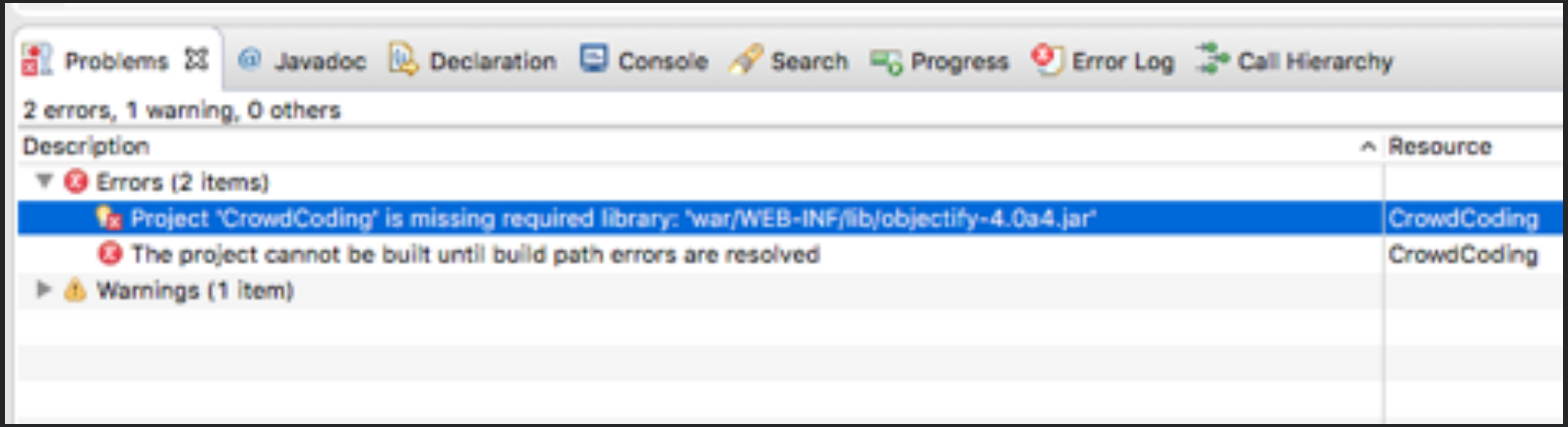
- Establishes relationship with user
- Important not to take user feel “stupid”
- Make the system take blame for errors
- Be positive, to encourage
- Provide helpful messages, not cute messages
- Avoid violent, negative, demeaning, threatening terms (e.g., illegal, invalid)

System Response Times

- 0.1second - reacting ***instantaneously***
  - requiring no special feedback except displaying result
  - limit for direct manipulation of objects in UI
- 1.0 second - ***freely*** navigating commands
  - noticeable delay, limit for keeping user's flow of thought uninterrupted
- 10 seconds - keeping users ***attention***
  - limit for keeping user's attention focus in UI
  - longer delays create task breaks
- [Nielsen, Usability Engineering, 1993]

Show Users How to Fix Errors

- Good: detecting user errors
- Better: directly showing how errors can be fixed
- (Best: using constraints to prevent errors from ever occurring)



2 errors, 1 warning, 0 others

Description	Resource
▼ Errors (2 items)	
Project 'CrowdCoding' is missing required library: 'war/WEB-INF/lib/objectify-4.0a4.jar'	CrowdCoding
The project cannot be built until build path errors are resolved	CrowdCoding
▶ Warnings (1 item)	

Adding Constraints to Block Errors

- Add specific constraints on actions
- e.g. forcing formatting in form fields
- Separate controls/fields so that those which are easily confused are far apart
- Separate items into different screens or modules

Undo

- Having an option to undo actions is one of the most powerful mechanisms to mitigate errors.
- However, this is not always possible, e.g. sending an email.

Norman's Key Design Principles

1. Put the knowledge required to to operate the technology in the world

2. Use the power of natural and artificial constraints

3. Bridge the two Gulfs: the Gulf of Execution and the Gulf of Evaluation

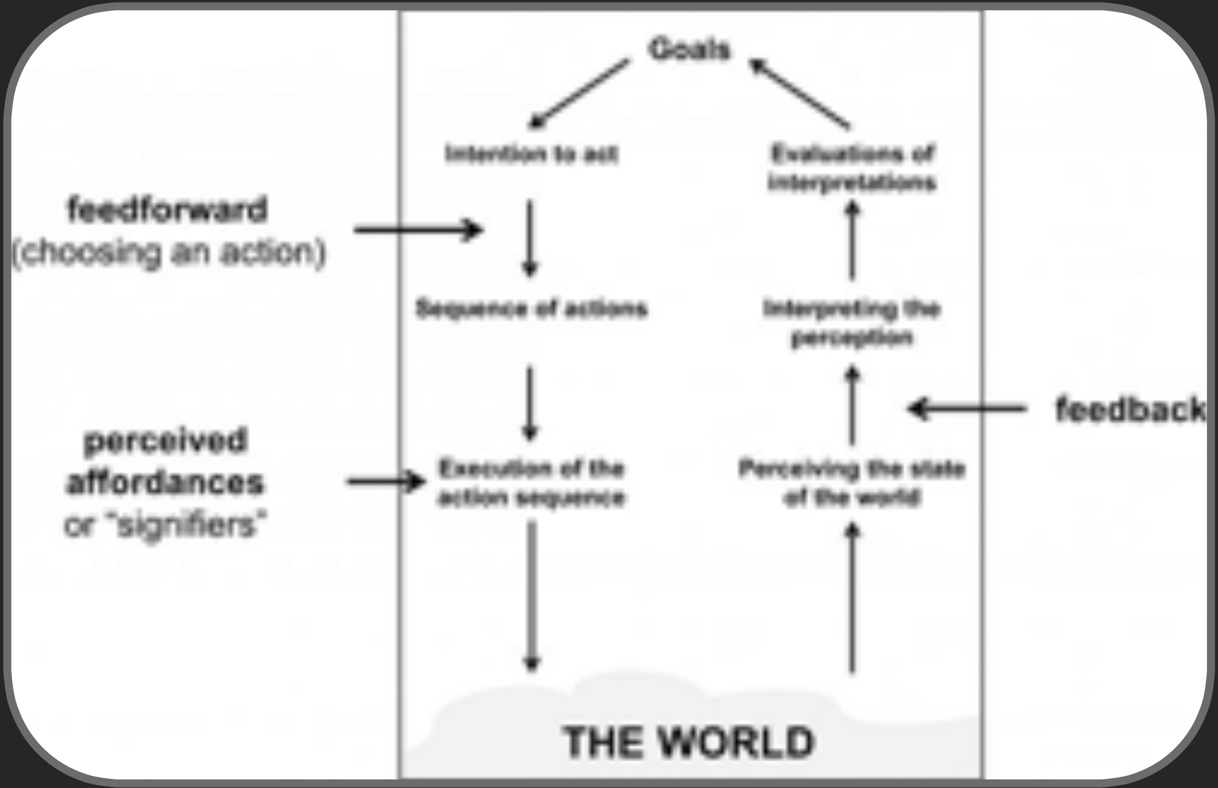
    - Execution: Make options readily available

    - Evaluation: Provide Feedback

Direct Manipulation

Motivation

- User is trying to do a task, manipulating a [model] of world
- Hard to plan out long sequence of actions in advance
- Gulf of execution: hard to know if took correct action
- Gulf of evaluation: hard to understand if successfully manipulated world
- Hard to compare hidden world to desired world

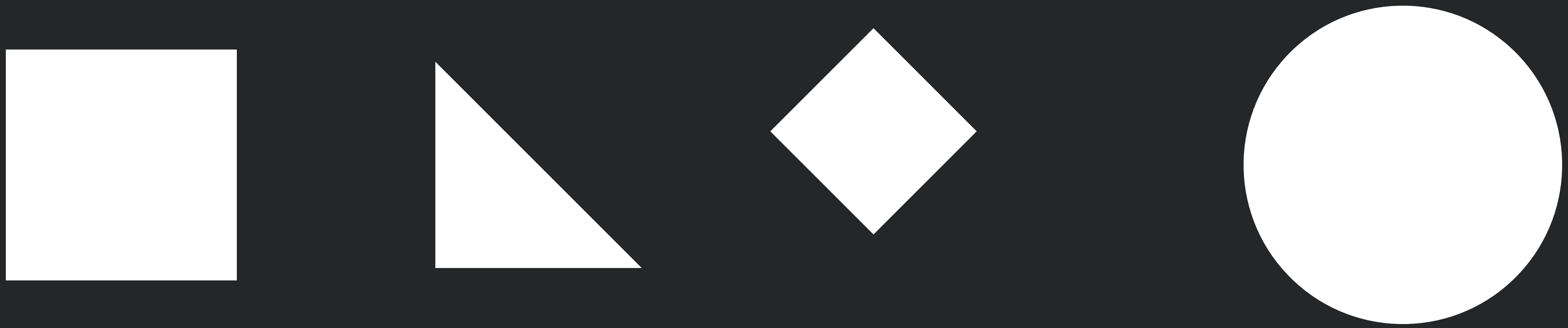


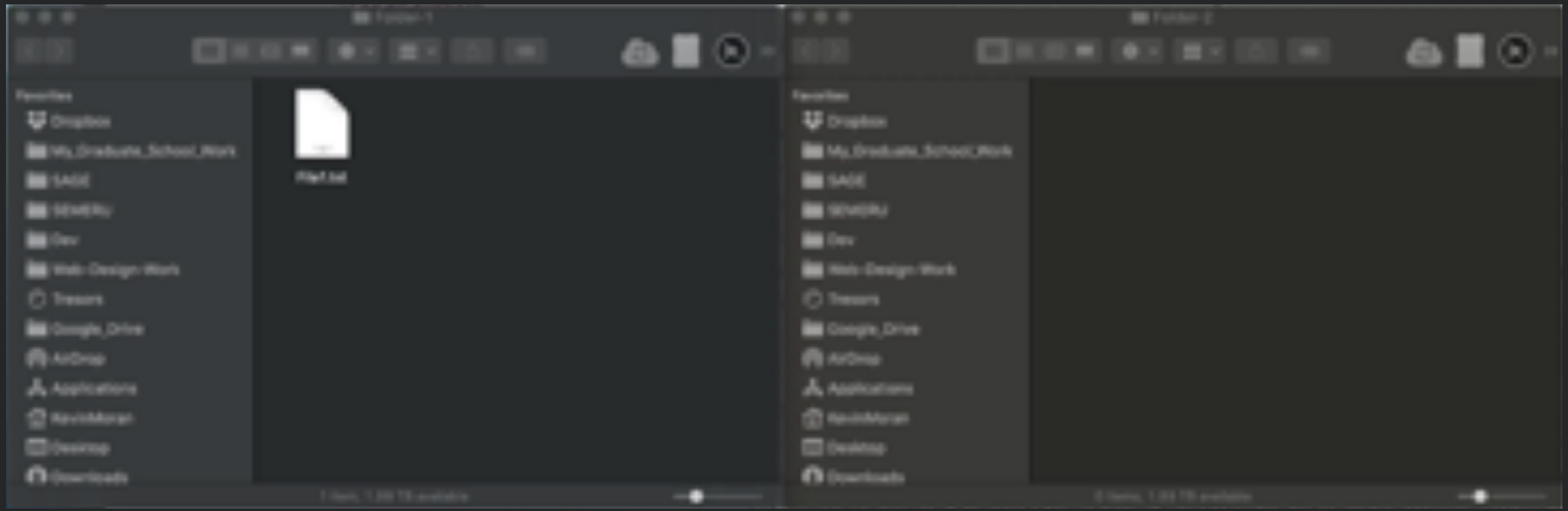
Key Questions

- What is the cost of an error?
  - Is it low cost or high cost?
  - Is it undoable?
- What feedback is necessary for user to realize the system is not in the desired state?

Direct Manipulation

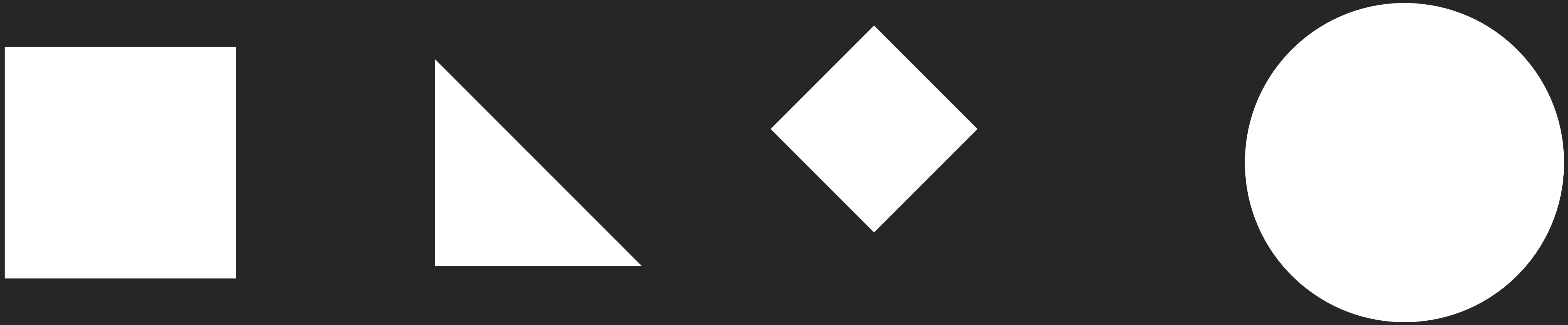
- “Rapid incremental reversible operations whose impact on the objects of interest is immediately visible” (Shneiderman, 1982)

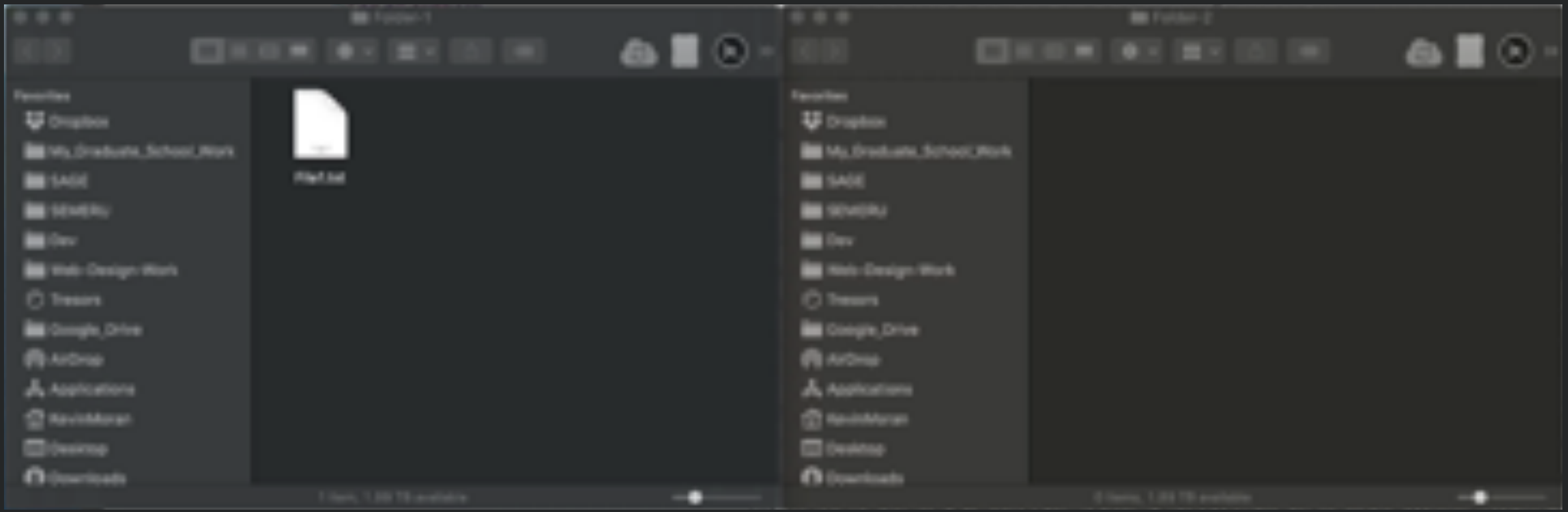




Direct Manipulation

- “Rapid incremental reversible operations whose impact on the objects of interest is immediately visible” (Shneiderman, 1982)





Direct Manipulation Characteristics

- Continuous Representation of the Object of Interest
- Physical Actions instead of complex syntax
- Continuous feedback and reversible, incremental actions

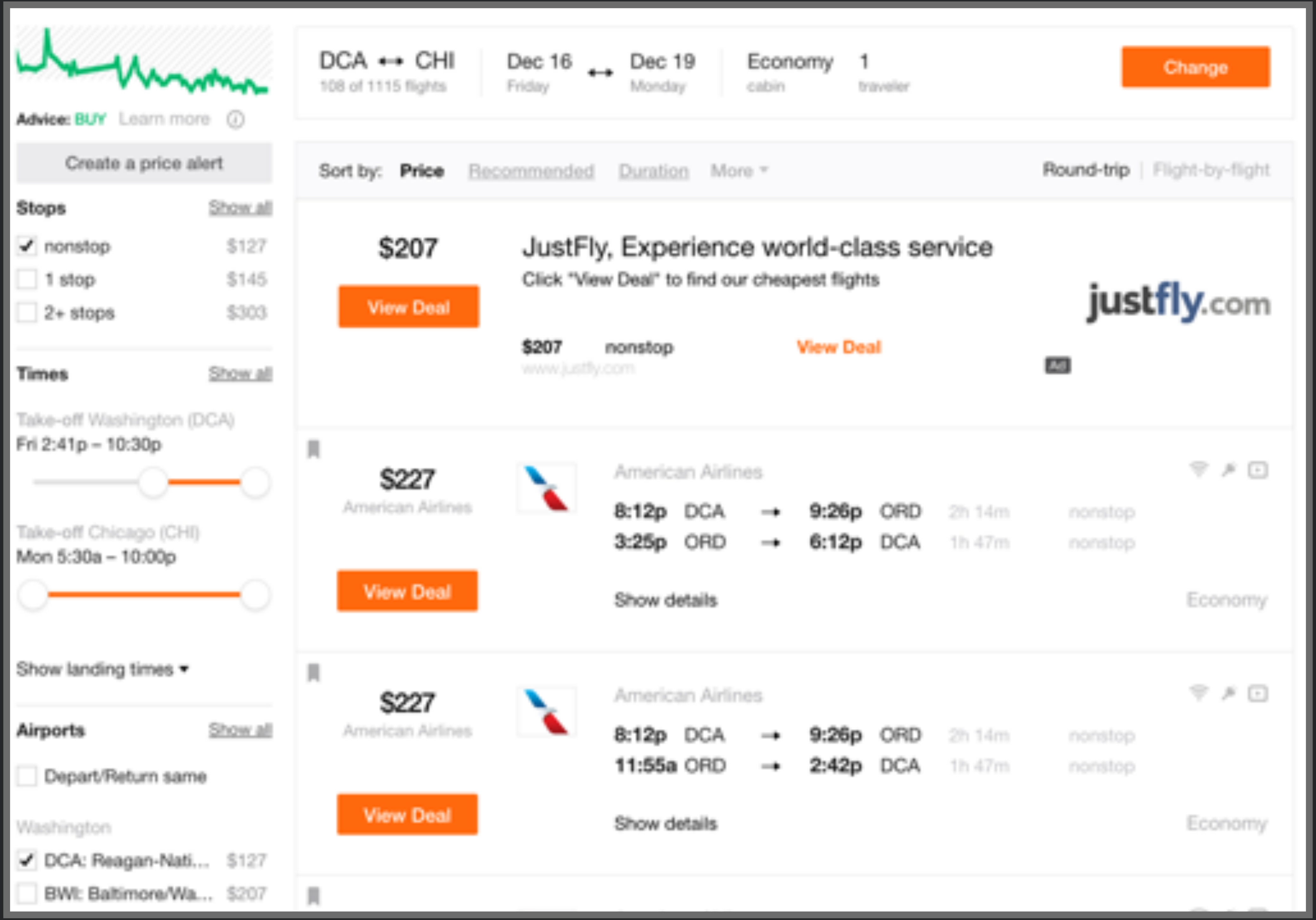
Benefits

- Supports exploration
  - Don’t plan long sequence of actions: pick an action, try it, can change mind if want to do something else instead
- Provides immediate feedback
  - Can quickly see what outcome of actions are in manipulating the world
  - Easy to compare desired state of the world to actual state of the world

Drawbacks

- Only a small Number of Objects on screen at once
- It can be physically demanding on the user
- Can be relatively slow
    - If the user needs to perform a large number of actions, it may be impractical
- Repetitive tasks are not well supported
    - e.g. can be better for novices to learn, but harder to experts to exploit
- Some gestures can be error prone

Example - Kayak



Advice: BUY Learn more ⓘ

Create a price alert

Stops Show all

☑ nonstop $127

□ 1 stop $145

□ 2+ stops $303

Times Show all

Take-off Washington (DCA)
Fri 2:41p – 10:30p

Take-off Chicago (CHI)
Mon 5:30a – 10:00p

Show landing times ▾

Airports Show all

□ Depart/Return same

Washington

☑ DCA: Reagan-Nati... $127

□ BWI: Baltimore/Wa... $207

DCA ↔ CHI
108 of 1115 flights

Dec 16 ↔ Dec 19
Friday Monday

Economy
cabin

1
traveler

Change

Sort by: **Price** Recommended Duration More ▾ Round-trip | Flight-by-flight

$207

View Deal

JustFly, Experience world-class service
Click "View Deal" to find our cheapest flights

justfly.com

$207 nonstop View Deal
www.justfly.com

Ad

$227
American Airlines

View Deal

American Airlines

8:12p	DCA	→	9:26p	ORD	2h 14m	nonstop
3:25p	ORD	→	6:12p	DCA	1h 47m	nonstop

Show details Economy

$227
American Airlines

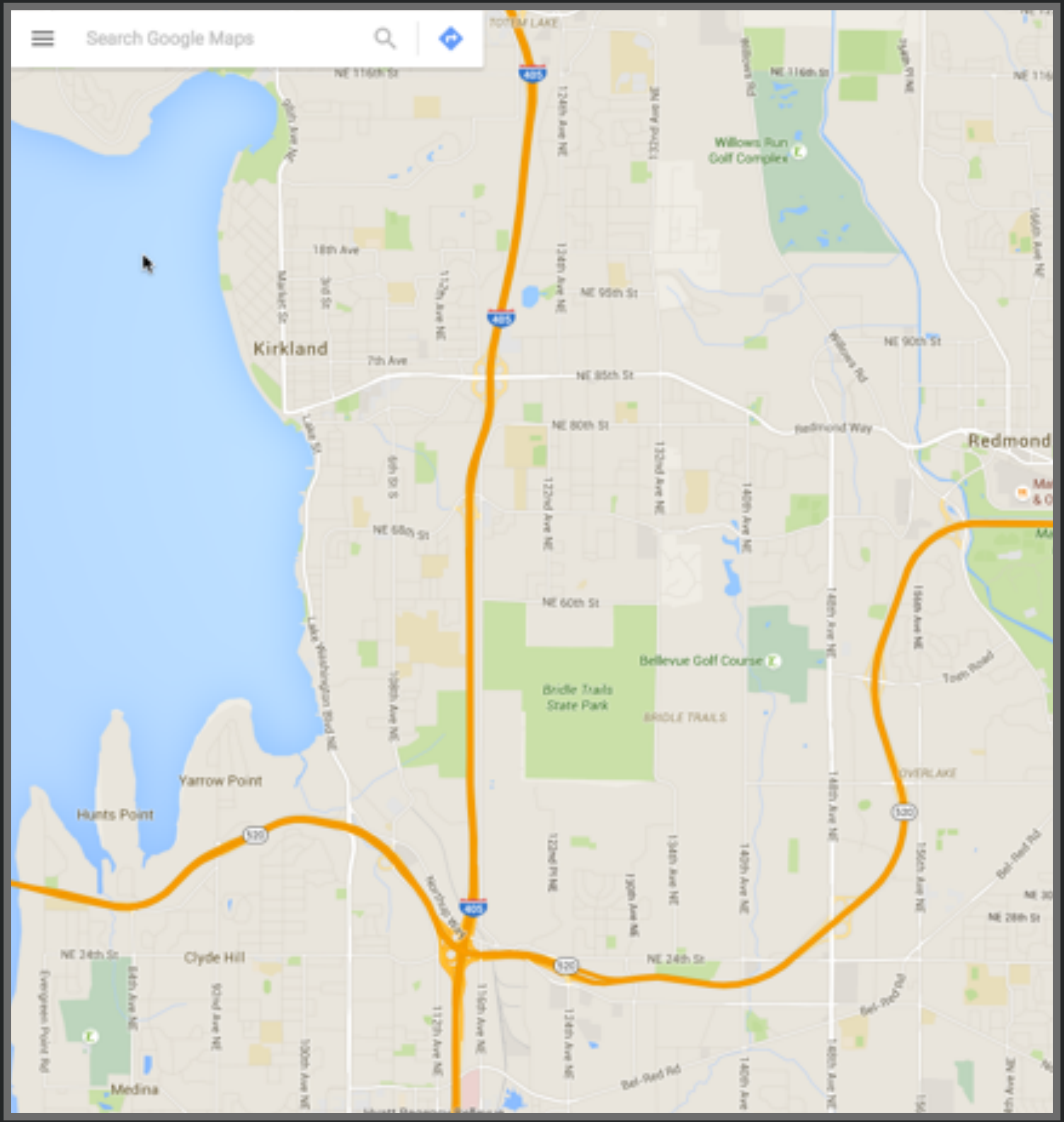
View Deal

American Airlines

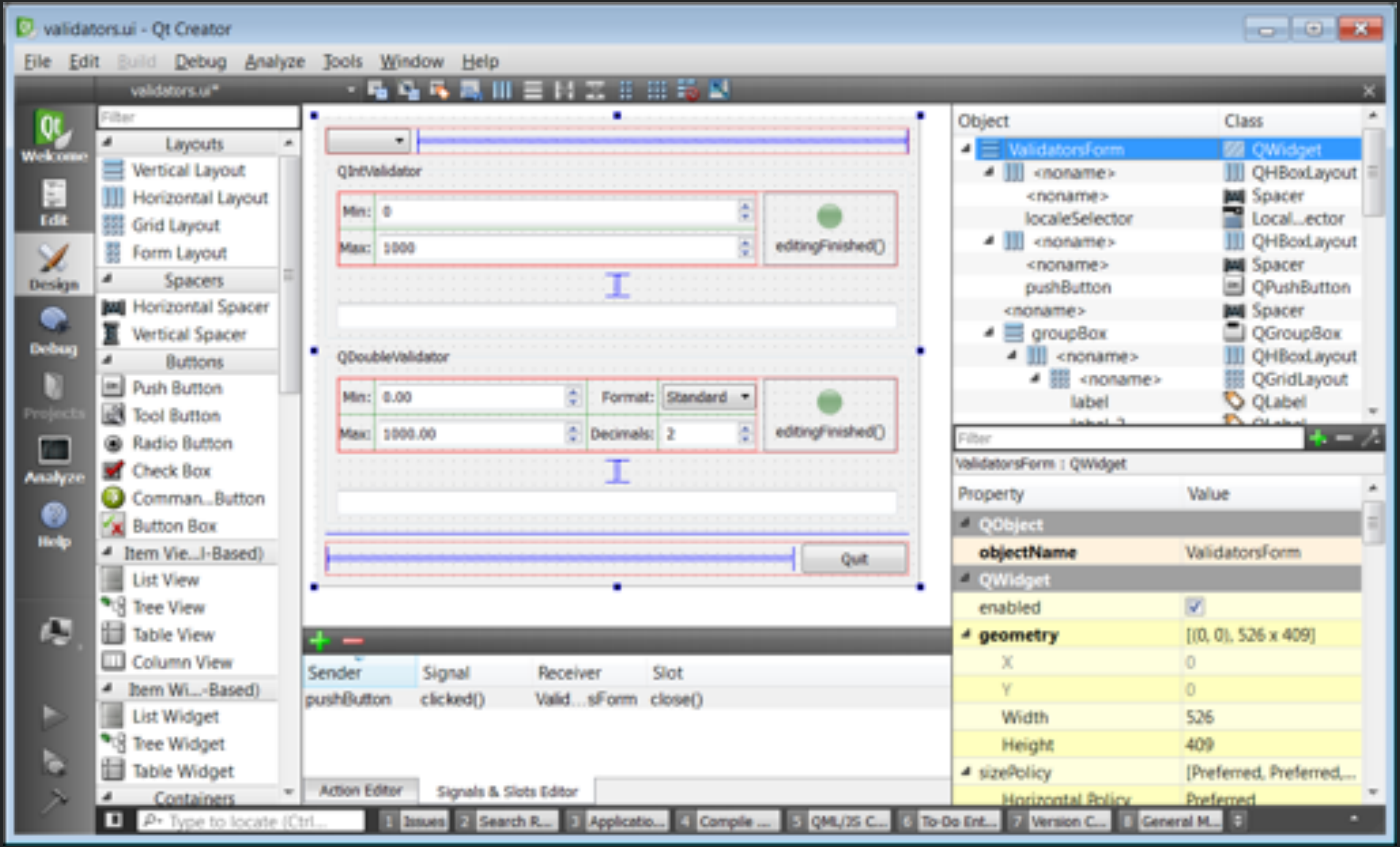
8:12p	DCA	→	9:26p	ORD	2h 14m	nonstop
11:55a	ORD	→	2:42p	DCA	1h 47m	nonstop

Show details Economy

Example - Google Maps



Example - GUI Builder



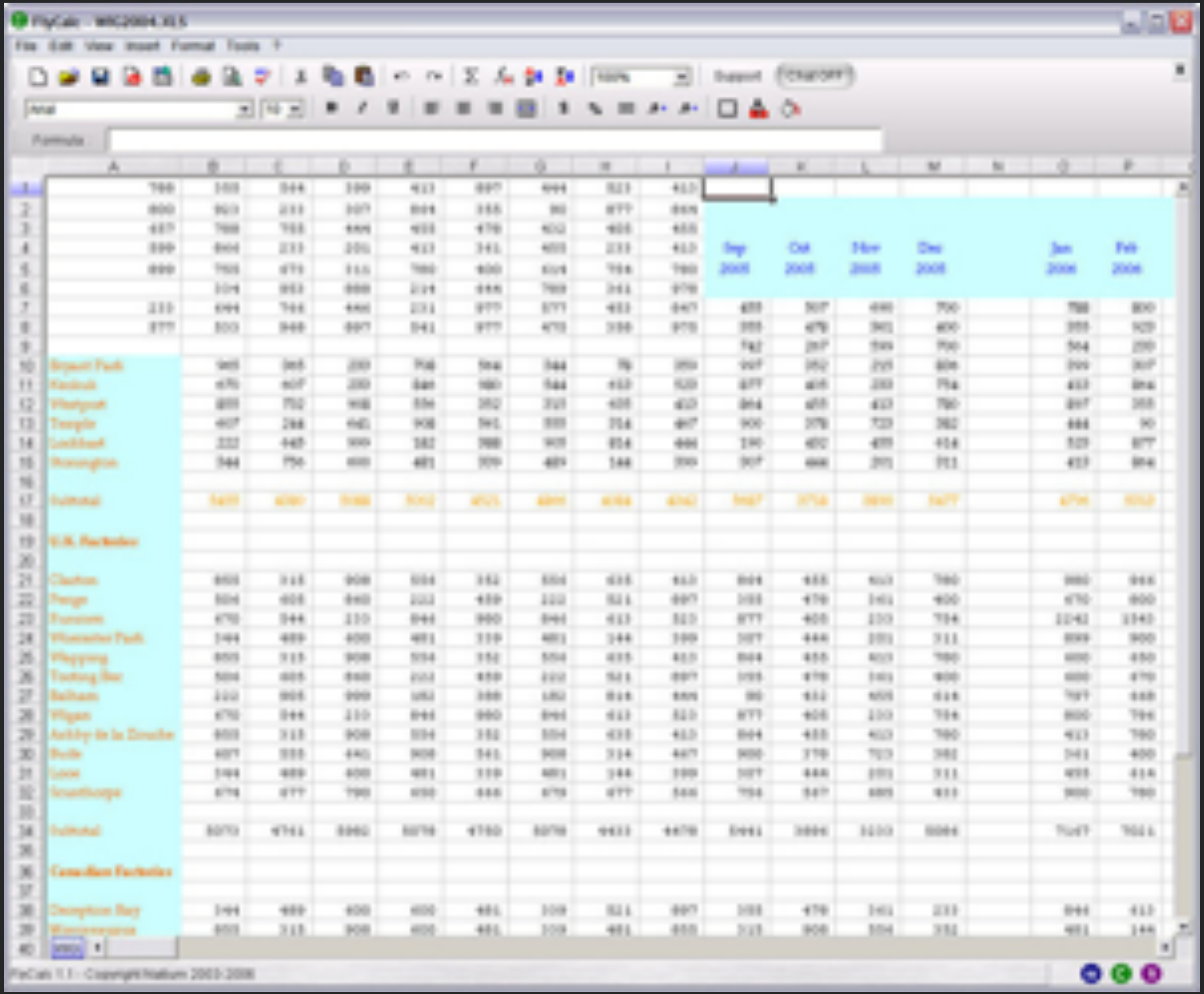
Object	Class
ValidatorsForm	QWidget
<noname>	QHBoxLayout
<noname>	Spacer
localeSelector	Local...ector
<noname>	QHBoxLayout
<noname>	Spacer
pushButton	QPushButton
<noname>	Spacer
groupBox	QGroupBox
<noname>	QHBoxLayout
<noname>	QGridLayout
label	QLabel
label_2	QLabel

Sender	Signal	Receiver	Slot
pushButton	clicked()	Valid...sForm	close()

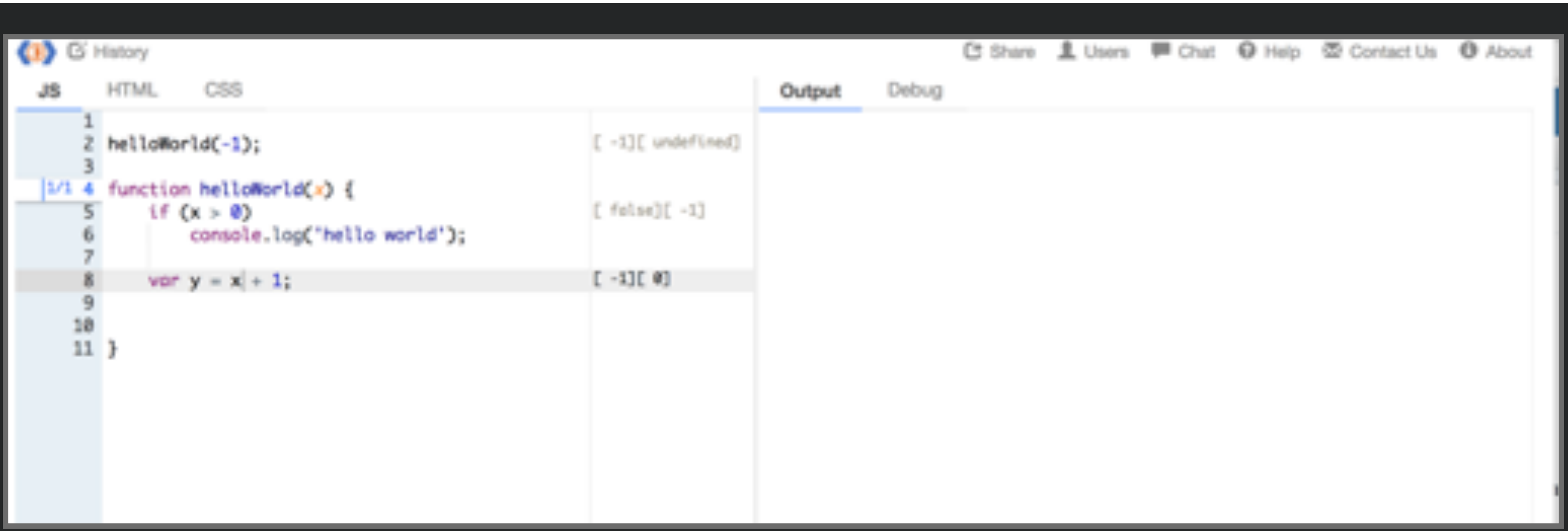
ValidatorsForm : QWidget

Property	Value
QObject	
objectName	ValidatorsForm
QWidget	
enabled	☑
geometry	[(0, 0), 526 x 409]
X	0
Y	0
Width	526
Height	409
sizePolicy	[Preferred, Preferred,...
Horizontal Policy	Preferred

Example - Spreadsheets



Example: Live Programming



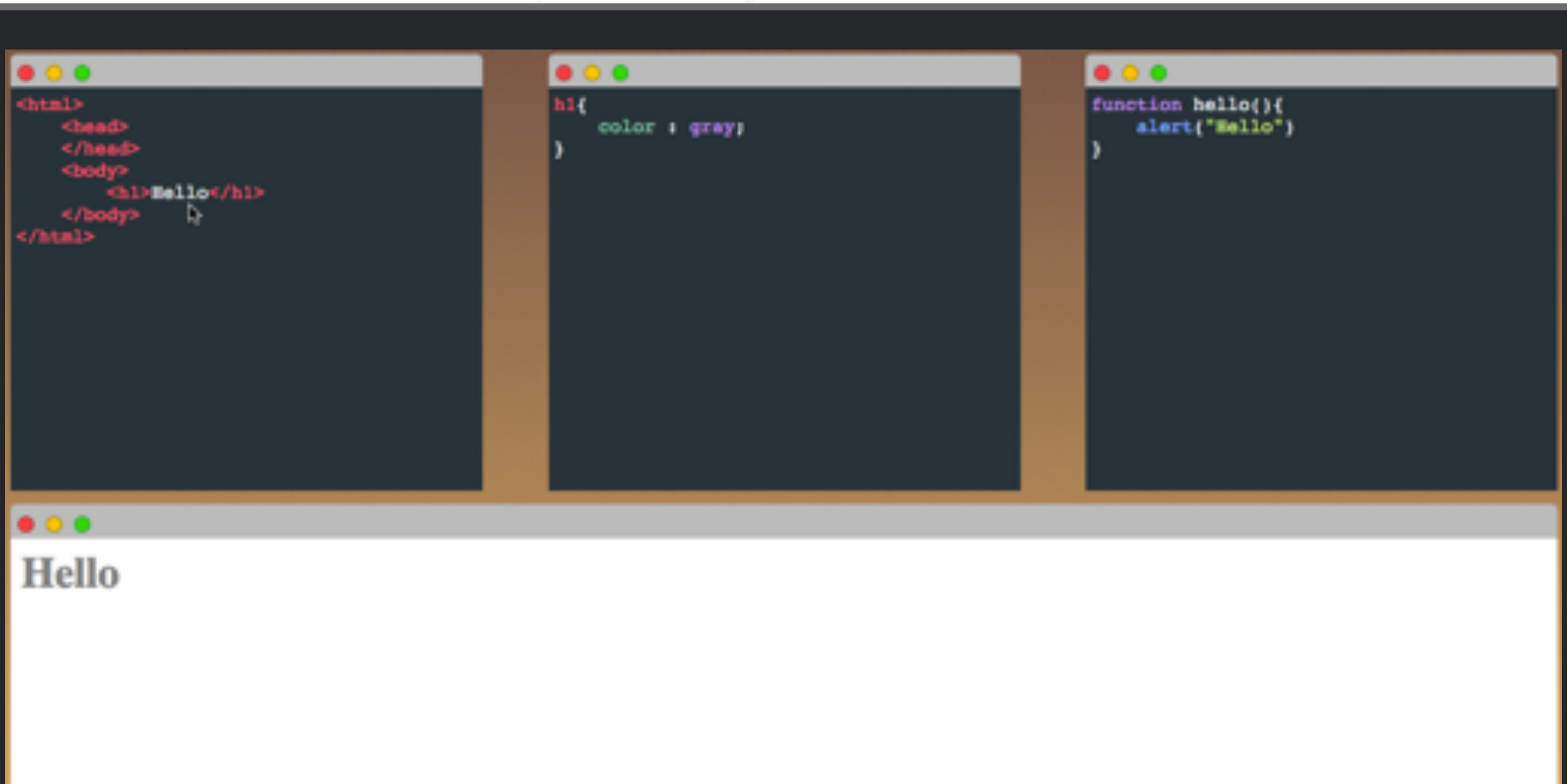
```
helloWorld(-1);                                   [ -1][ undefined]

function helloWorld(x) {
    if (x > 0)                                    [ false][ -1]
        console.log('hello world');

    var y = x + 1;                                [ -1][ 0]



}
```



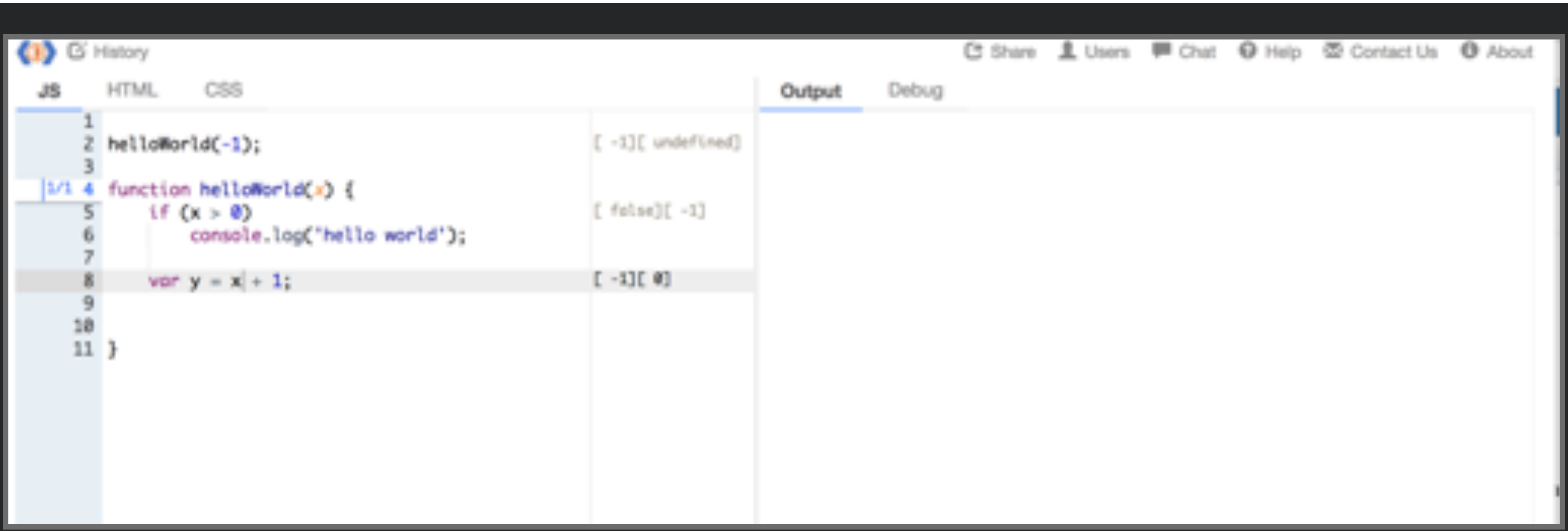
```html
<html>
    <head>
    </head>
    <body>
        <h1>Hello</h1>
    </body>
</html>
```

```css
h1{
    color : gray;
}
```

```javascript
function hello(){
    alert("Hello")
}
```

Hello

Example: Live Programming



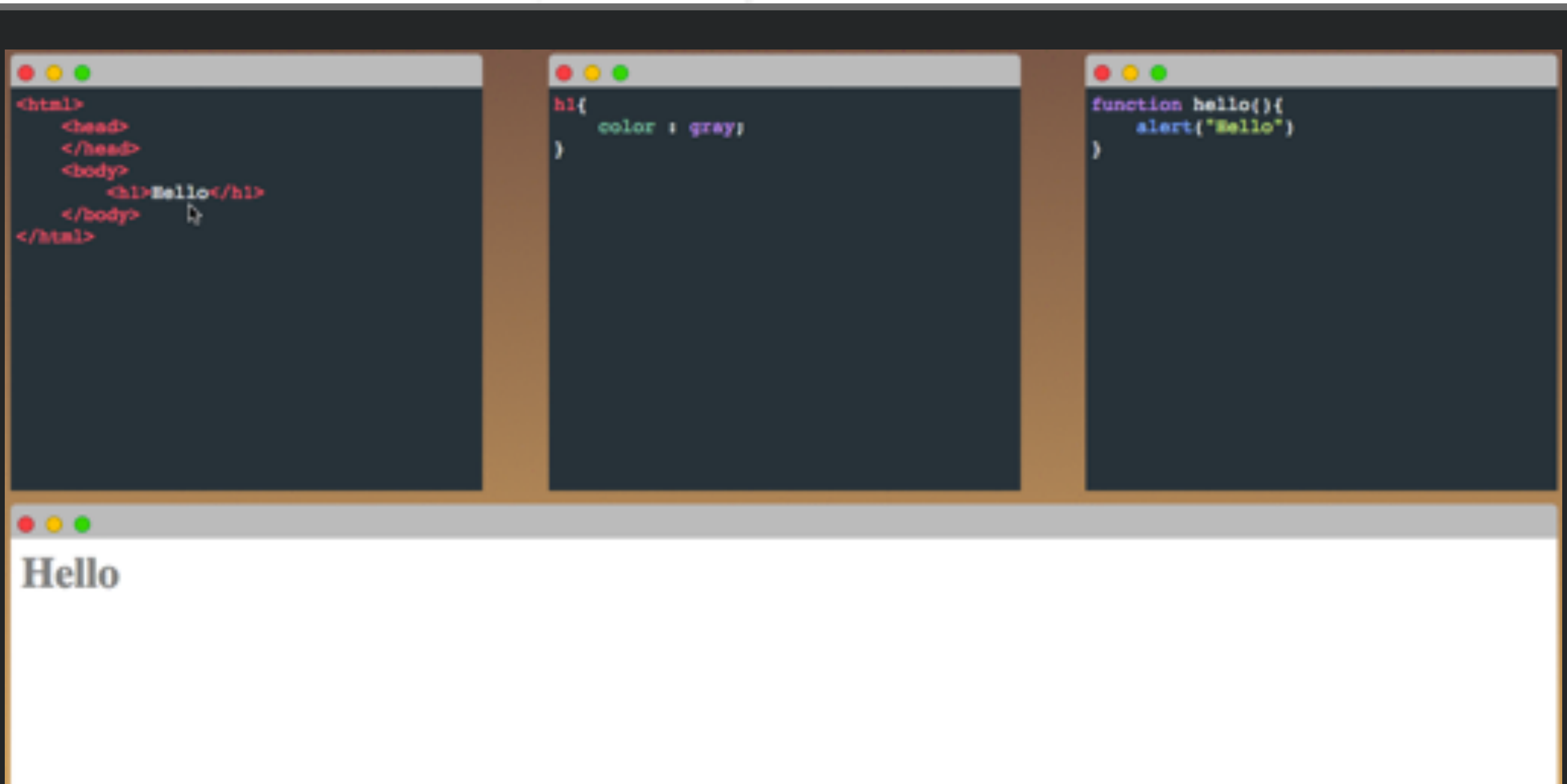
```
helloWorld(-1);                                  [ -1][ undefined]

function helloWorld(x) {
    if (x > 0)                                   [ false][ -1]
        console.log('hello world');

    var y = x + 1;                               [ -1][ 0]



}
```



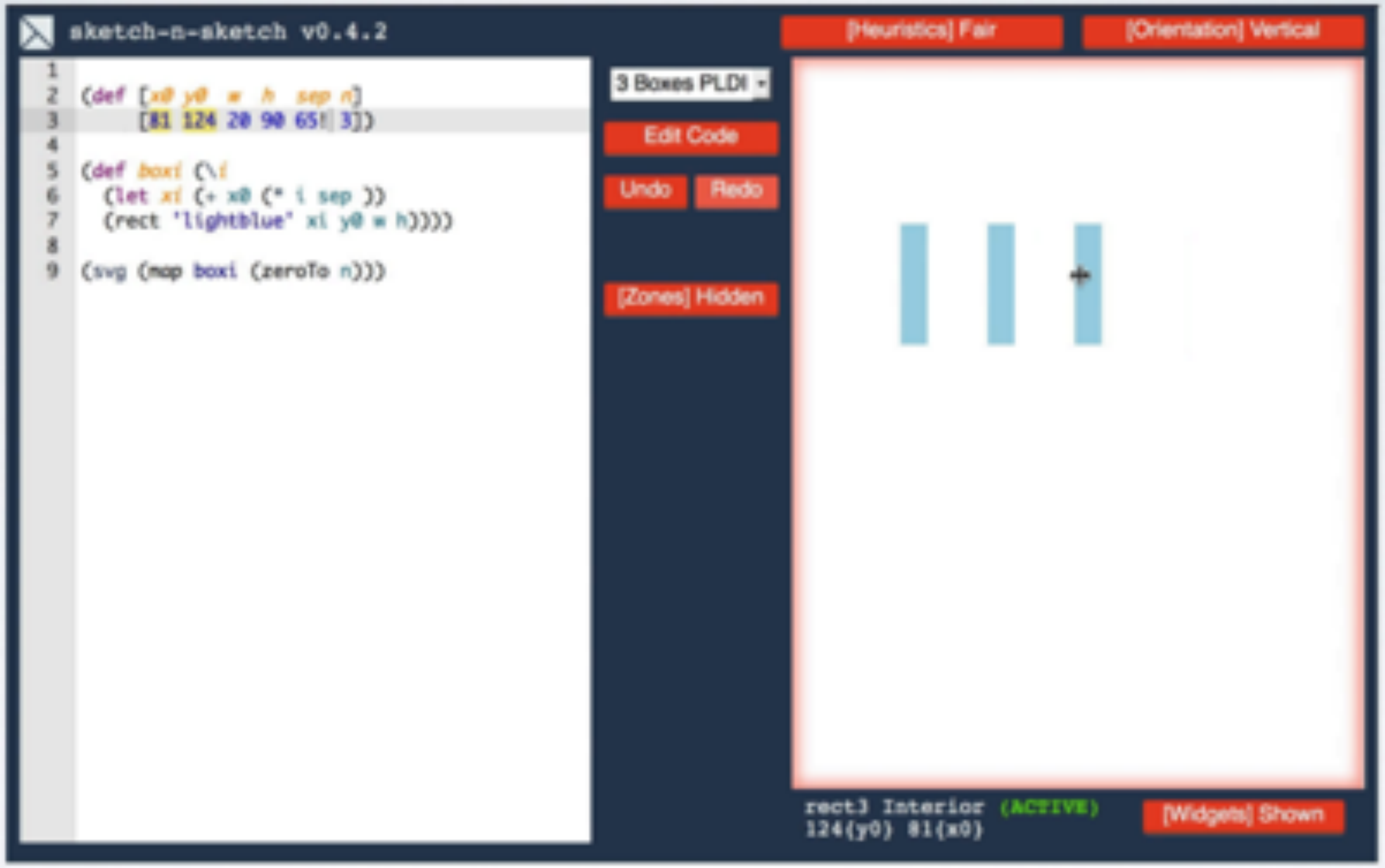
```
<html>
    <head>
    </head>
    <body>
        <h1>Hello</h1>
    </body>
</html>
```

```
h1{
    color : gray;
}
```

```
function hello(){
    alert("Hello")
}
```

Hello

Example: Edit Constants by Editing Output



```
(def [x0 y0  w  h  sep n]
     [81 124 20 90 65!| 3])

(def boxi (\i
  (let xi (+ x0 (* i sep ))
  (rect 'lightblue' xi y0 w h))))

(svg (map boxi (zeroTo n)))
```

```
rect3 Interior (ACTIVE)
124(y0) 81(x0)
```

Chugh et al. [PLDI '16]

7 Minute Break

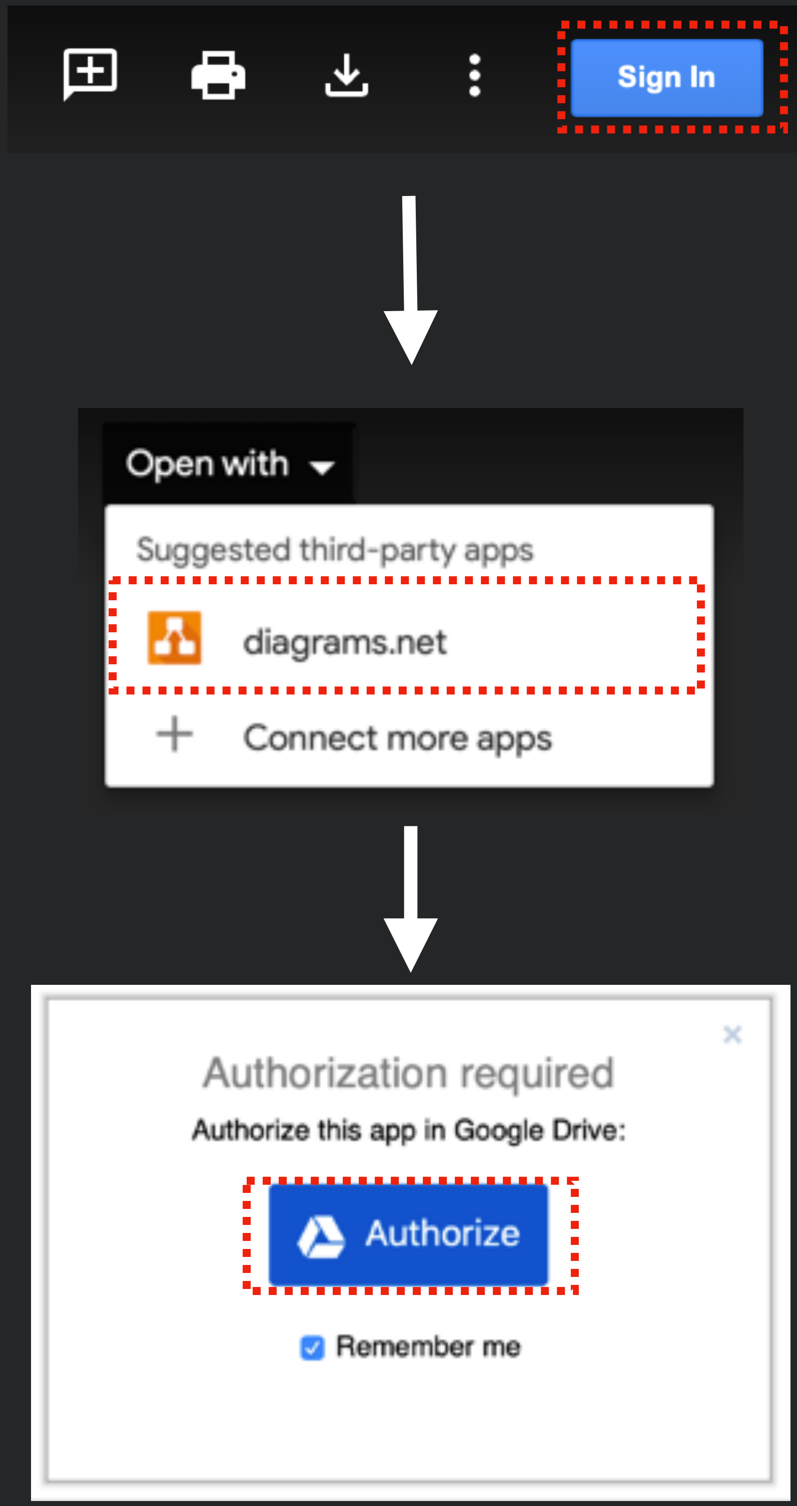
In-Class Activity

In Class Activity: Direct Manipulation Programming

- In groups of 2
  - Design a system for writing code through direct manipulation
    - Create sketches showing key screens
    - Should support
      - Standard programming language features (variables, conditionals, loops, functions)
      - Should make it faster and easier to make code changes
      - Should make it easier to get feedback on if program exhibits intended behavior

Deliverable: Sketches in the Google Drive Link

In Class Activity: Direct Manipulation Programming Interactions



Authorization required

Authorize this app in Google Drive:

In Class Activity: Direct Manipulation Programming

- In groups of 2
  - Design a system for writing code through direct manipulation
    - Create sketches showing key screens
    - Should support
      - Standard programming language features (variables, conditionals, loops, functions)
      - Should make it faster and easier to make code changes
      - Should make it easier to get feedback on if program exhibits intended behavior

Deliverable: Sketches in the Google Drive Link

Acknowledgements

- Slides adapted from Dr. Thomas Latoza's SWE 632 course