Preventing Error

SWE 632
Fall 2023

Administrivia

- HW5 due today
- HW6 due in 2 weeks

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. *Group Activity:* Designing a Direct Manipulation App
5. *Two Tech Talks*

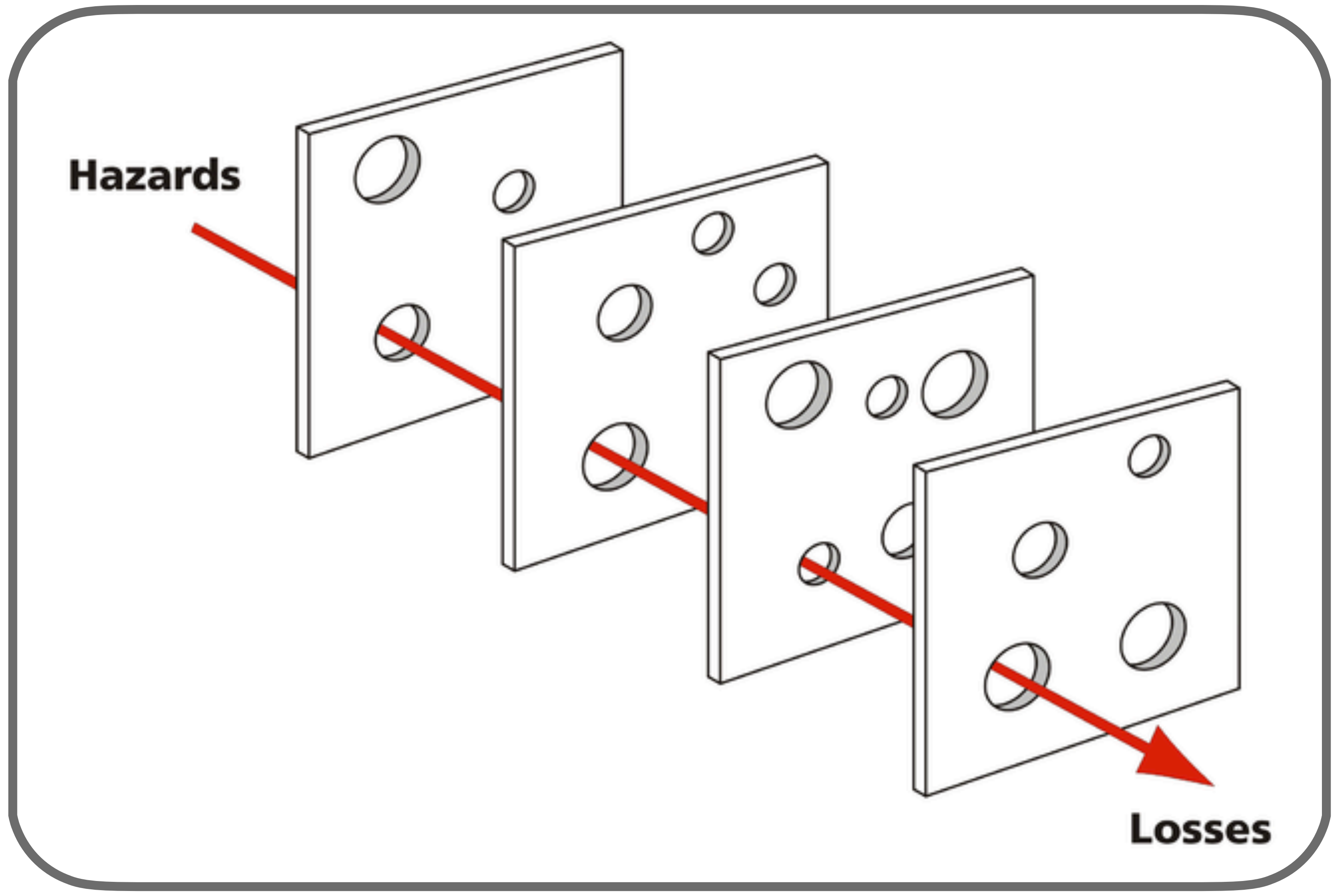
Human Error

What Causes Disasters?

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

Swiss Cheese Model

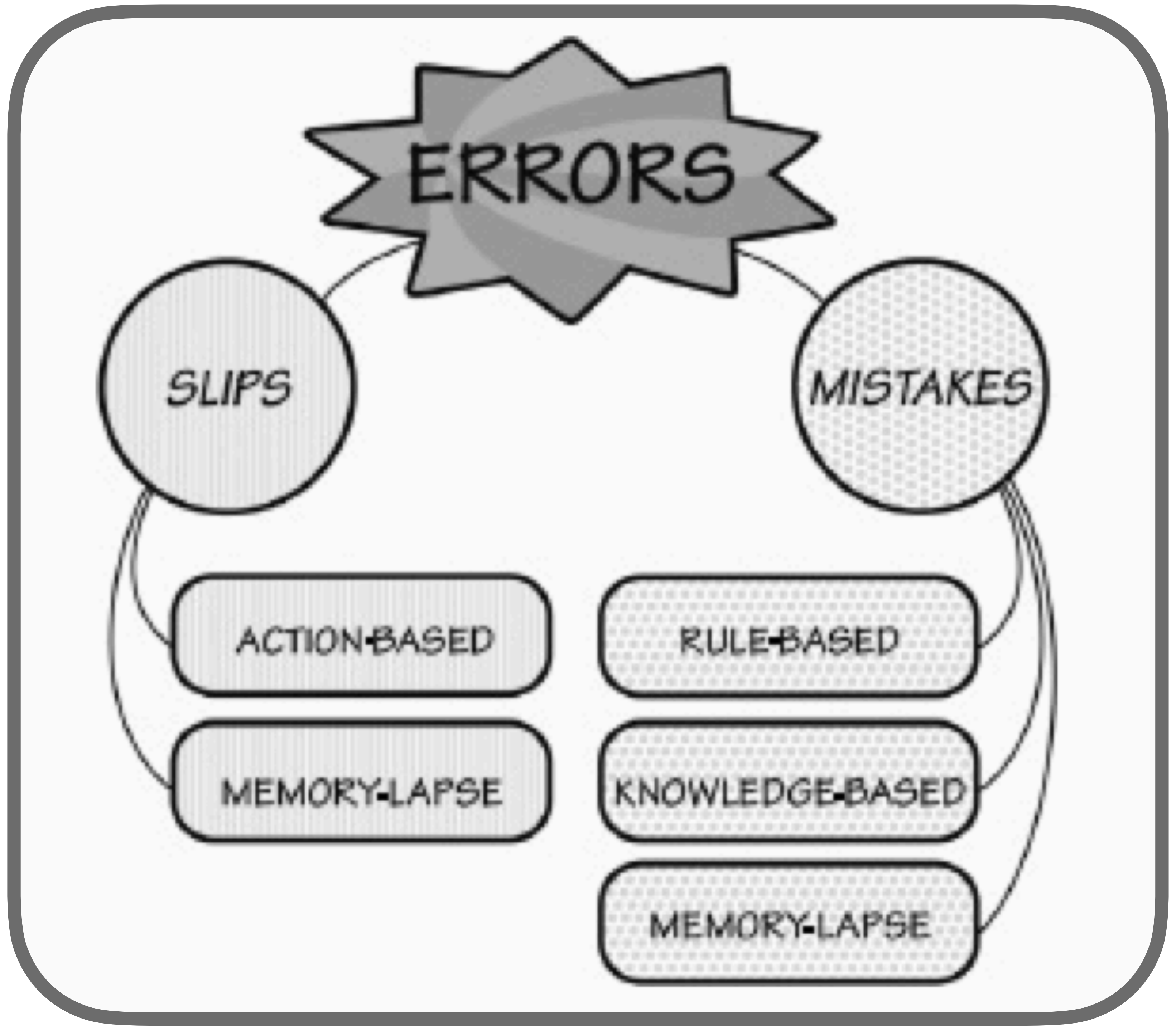
- Accidents must penetrate levels of system defenses
- Reduce accidents by
  - Adding more layers
  - Reduce the size and number of holes
  - Alert users when holes line up



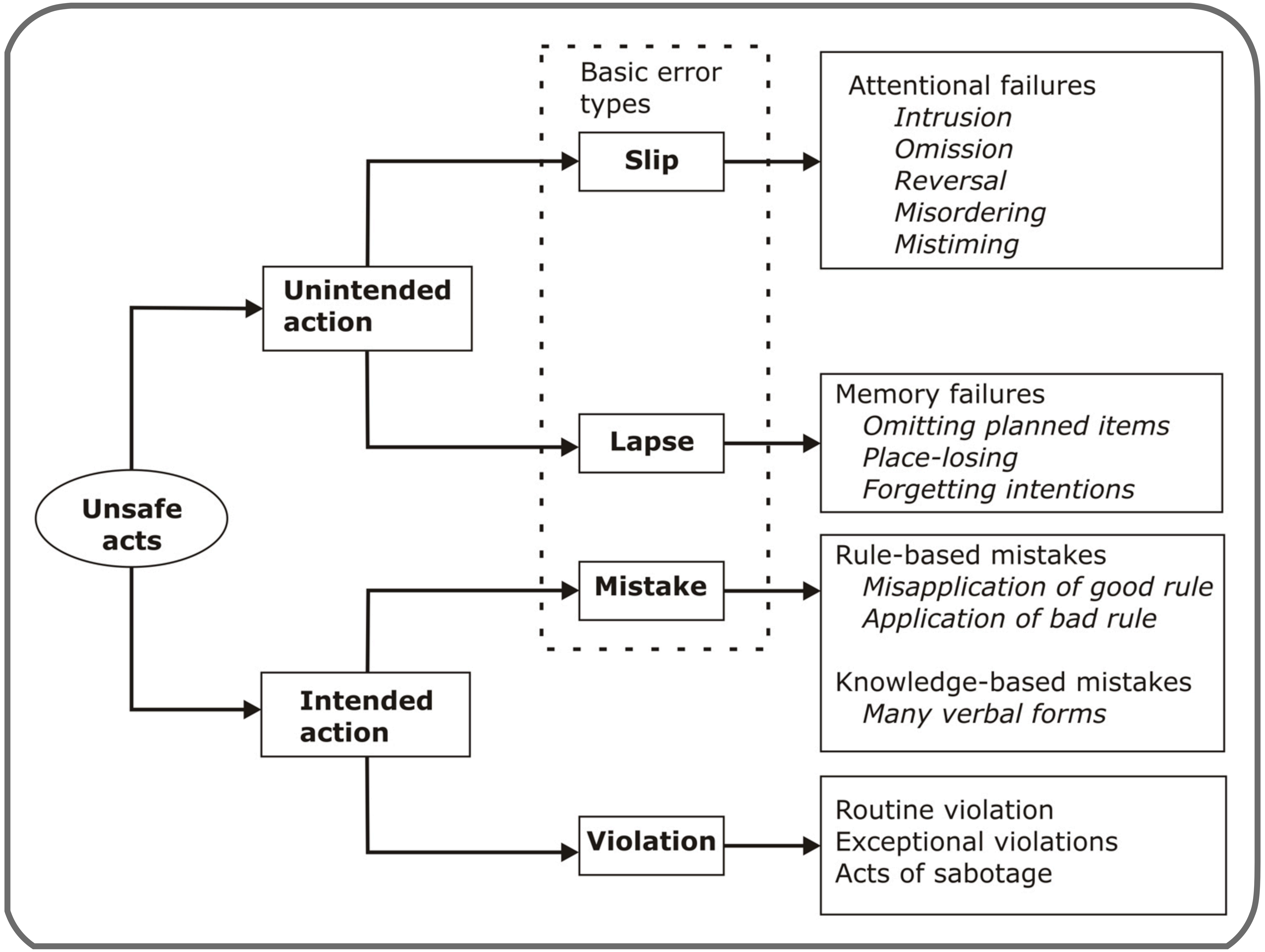
Root Cause Analysis

- 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.

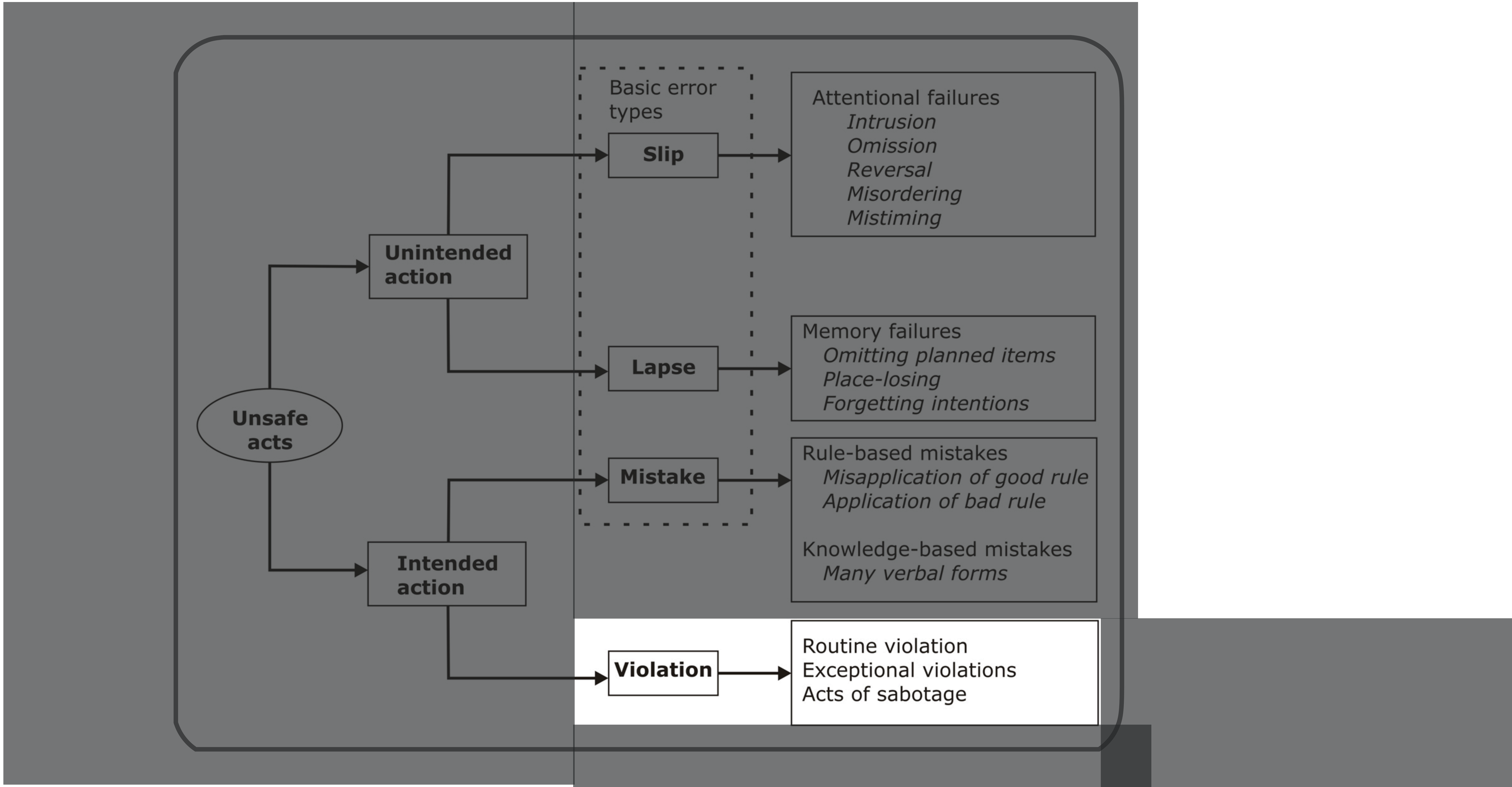
Psychological Types of Unsafe Acts



Psychological Types of Unsafe Acts



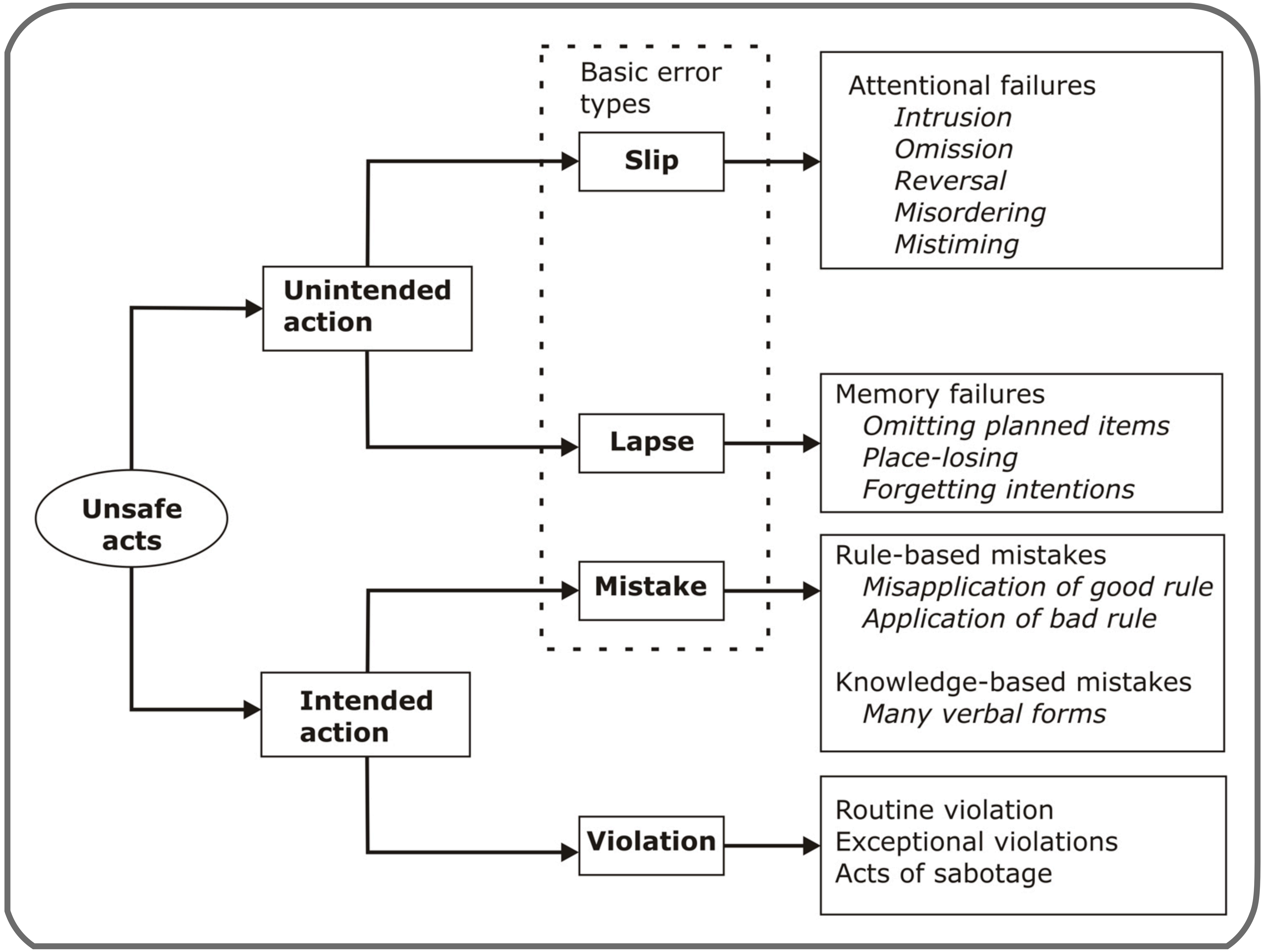
Psychological Types of Unsafe Acts



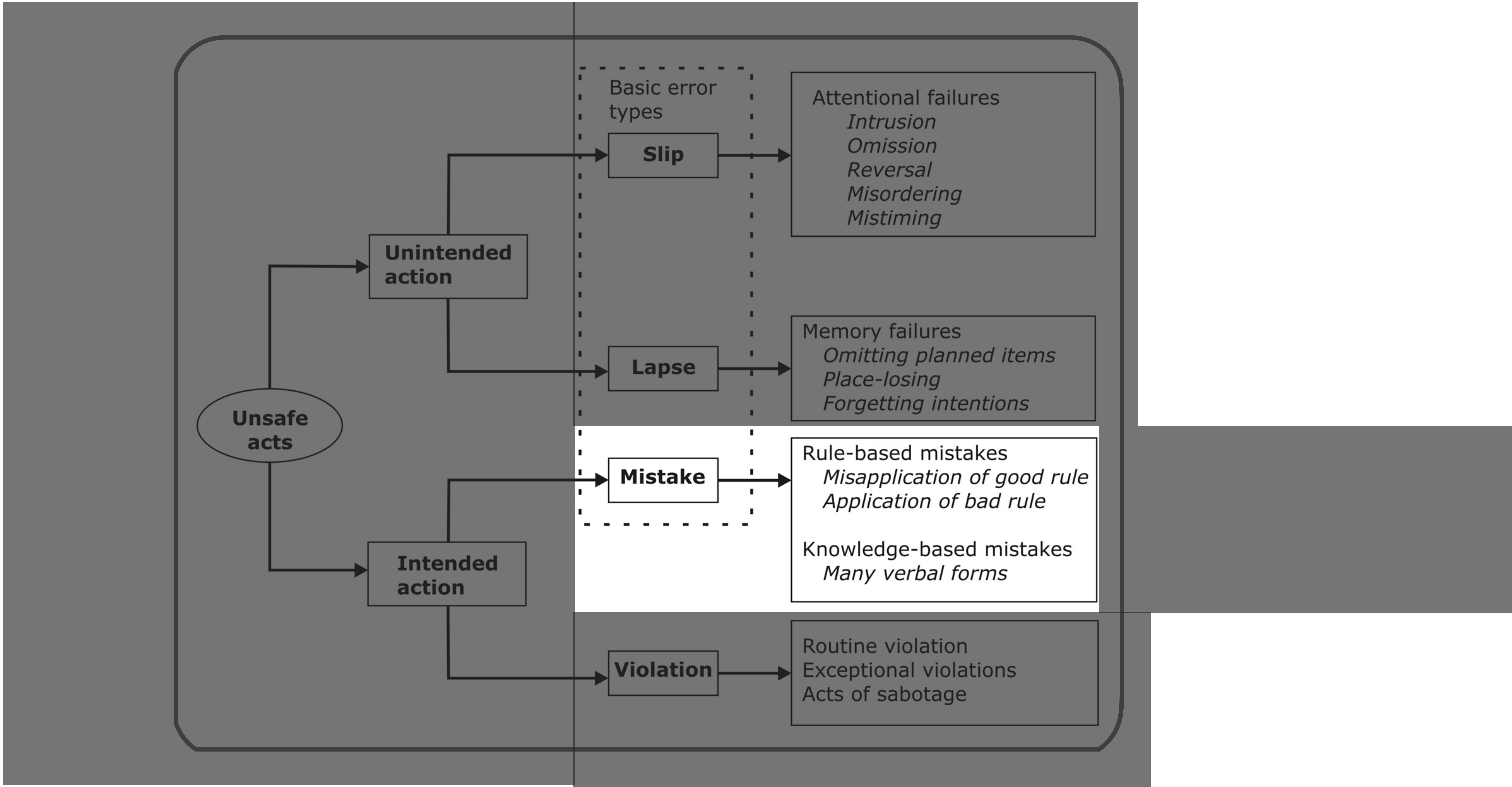
Deliberate Violations

- 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



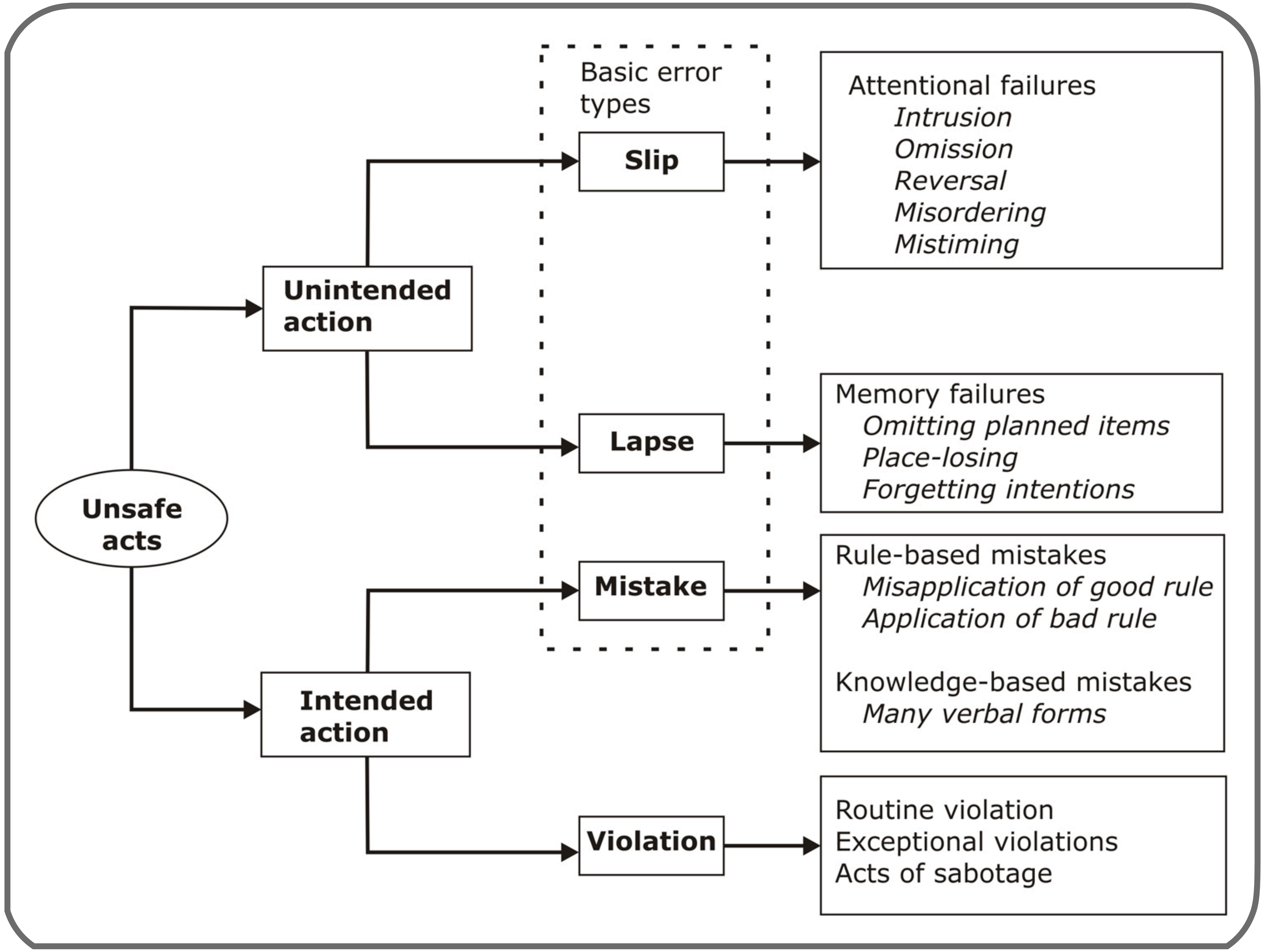
Psychological Types of Unsafe Acts



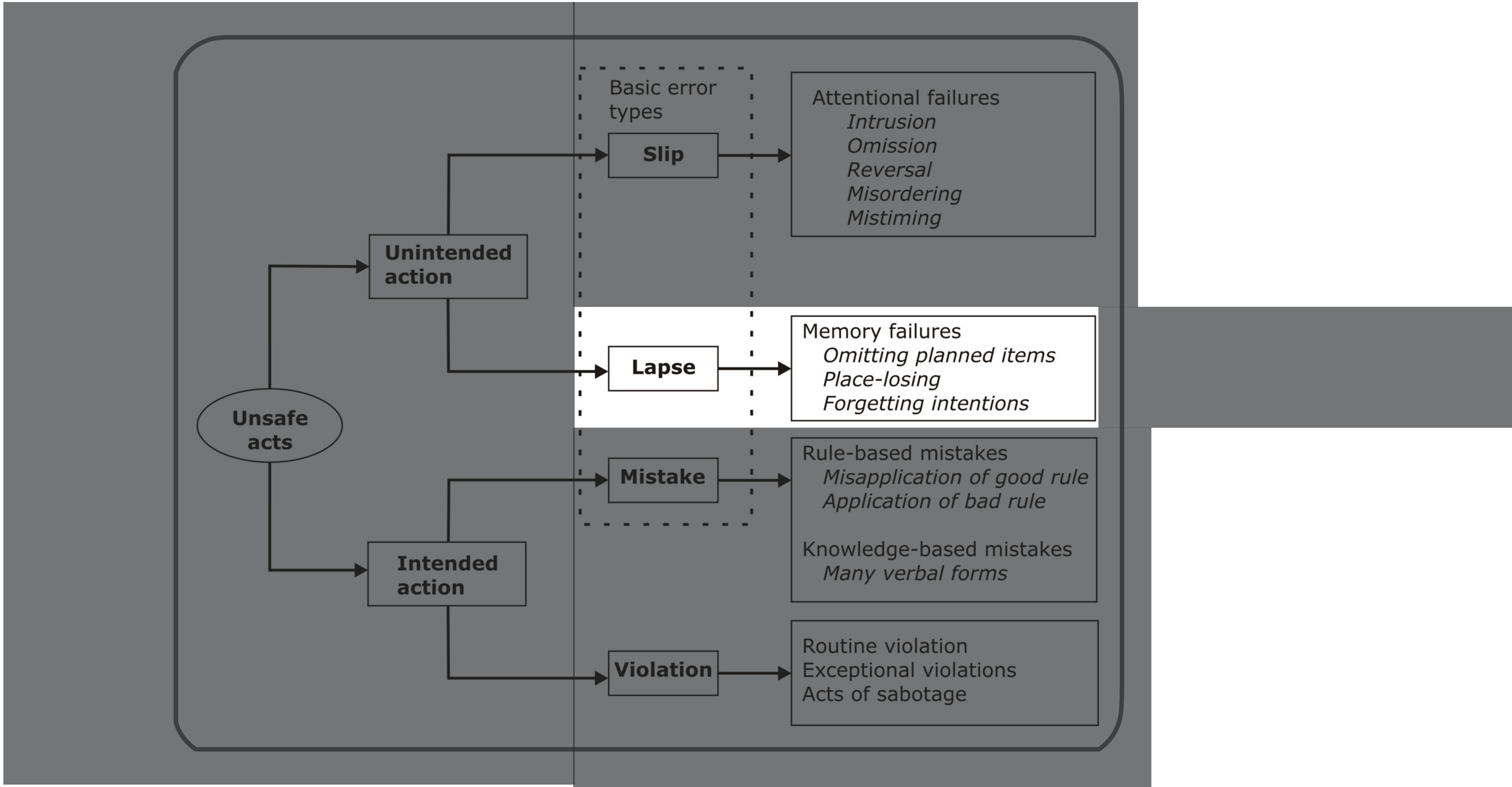
Mistakes

- 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



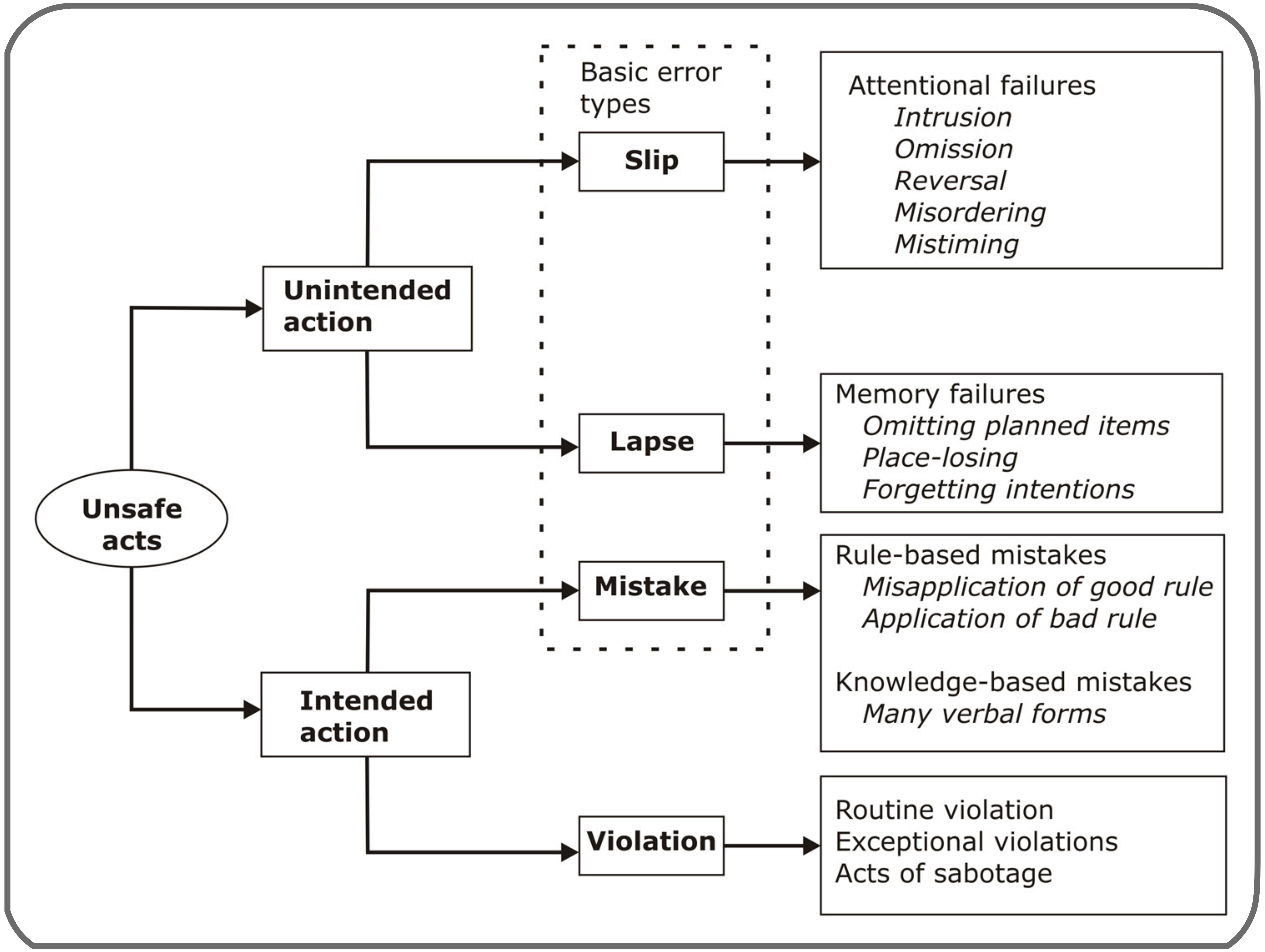
Psychological Types of Unsafe Acts



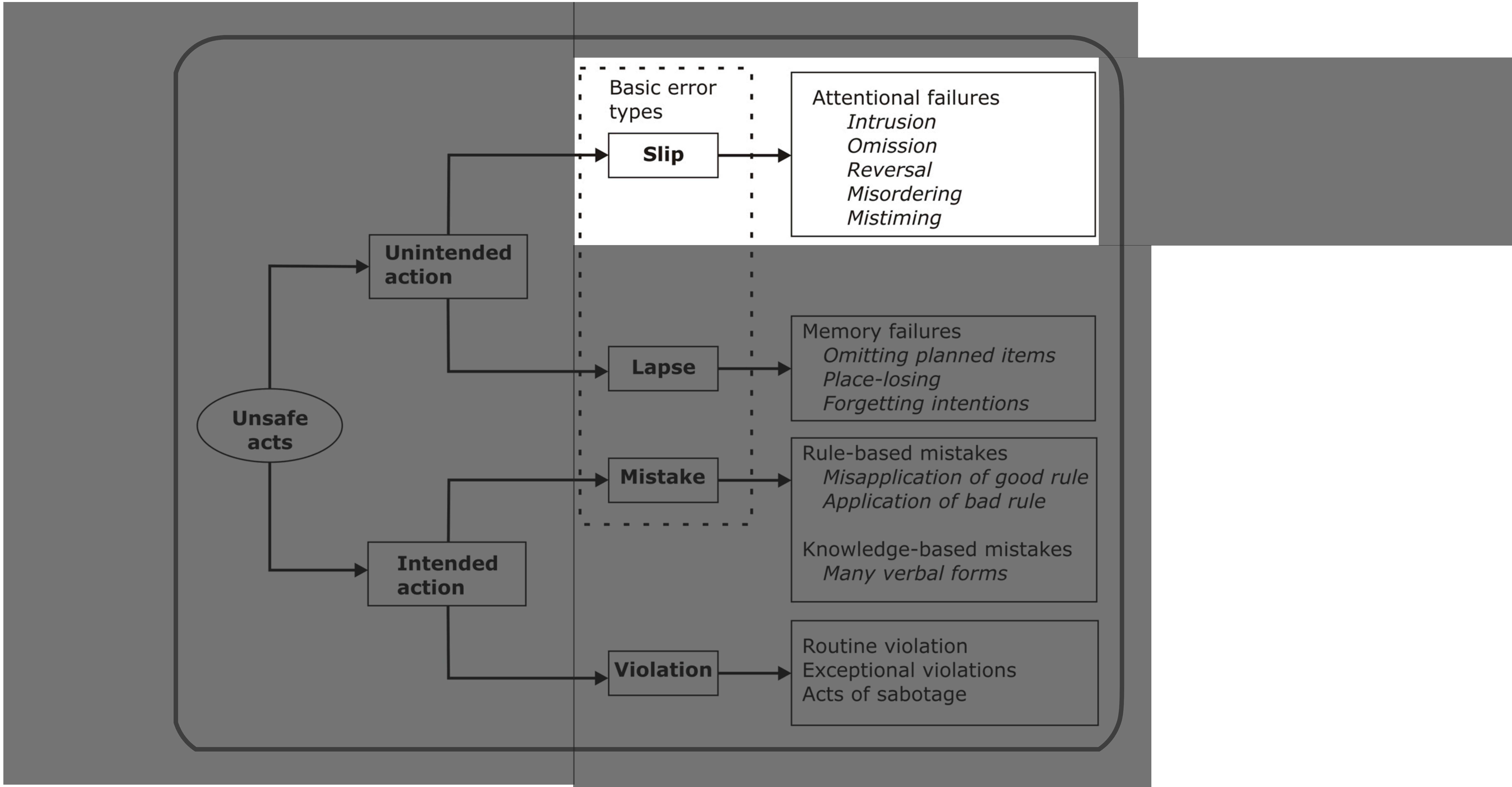
Memory Lapse

- 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



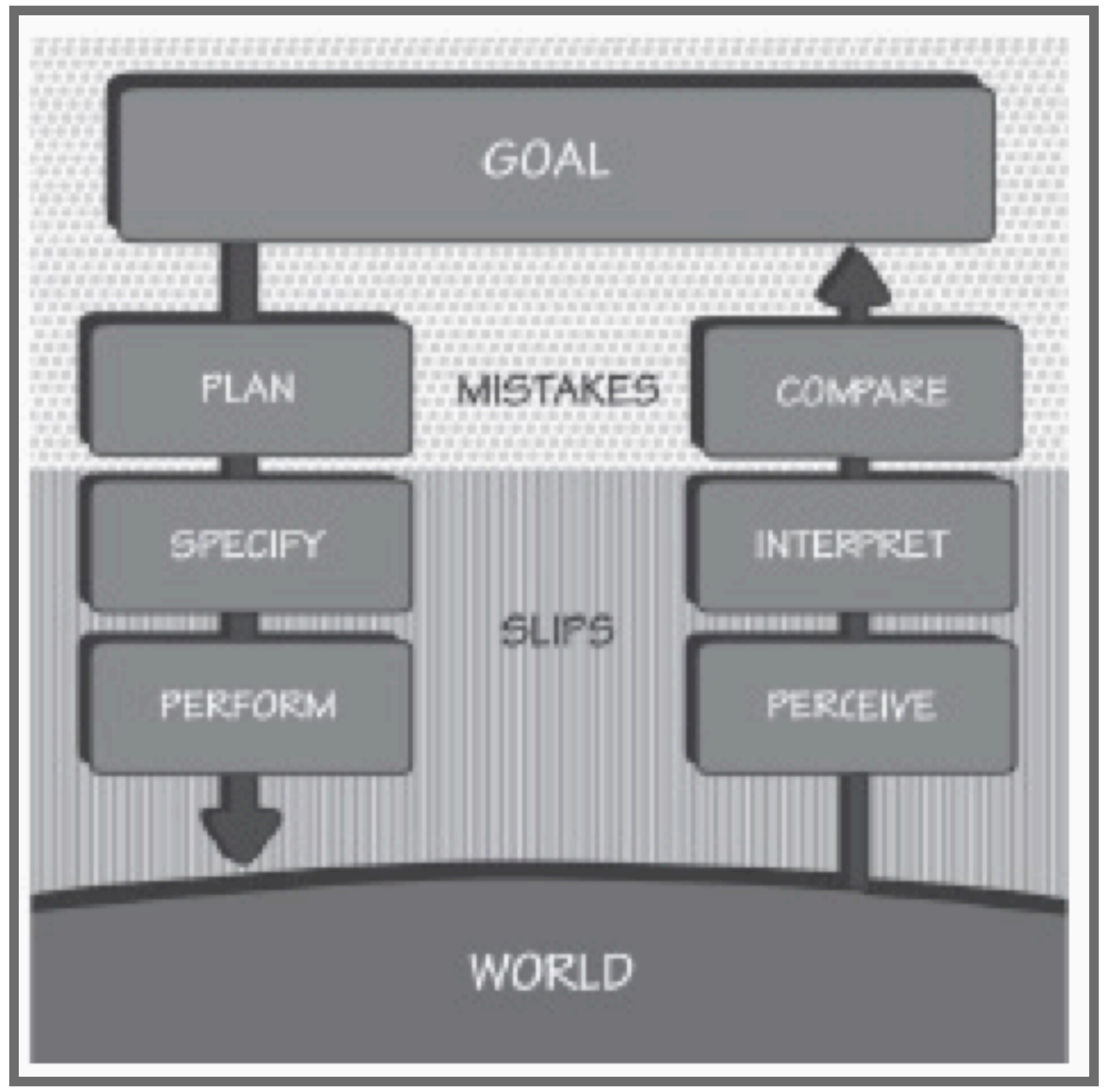
Psychological Types of Unsafe Acts



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



- ***Novices*** *are more likely to make mistakes than slips, and* ***experts*** *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

Activity

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

- What was the underlying cause?

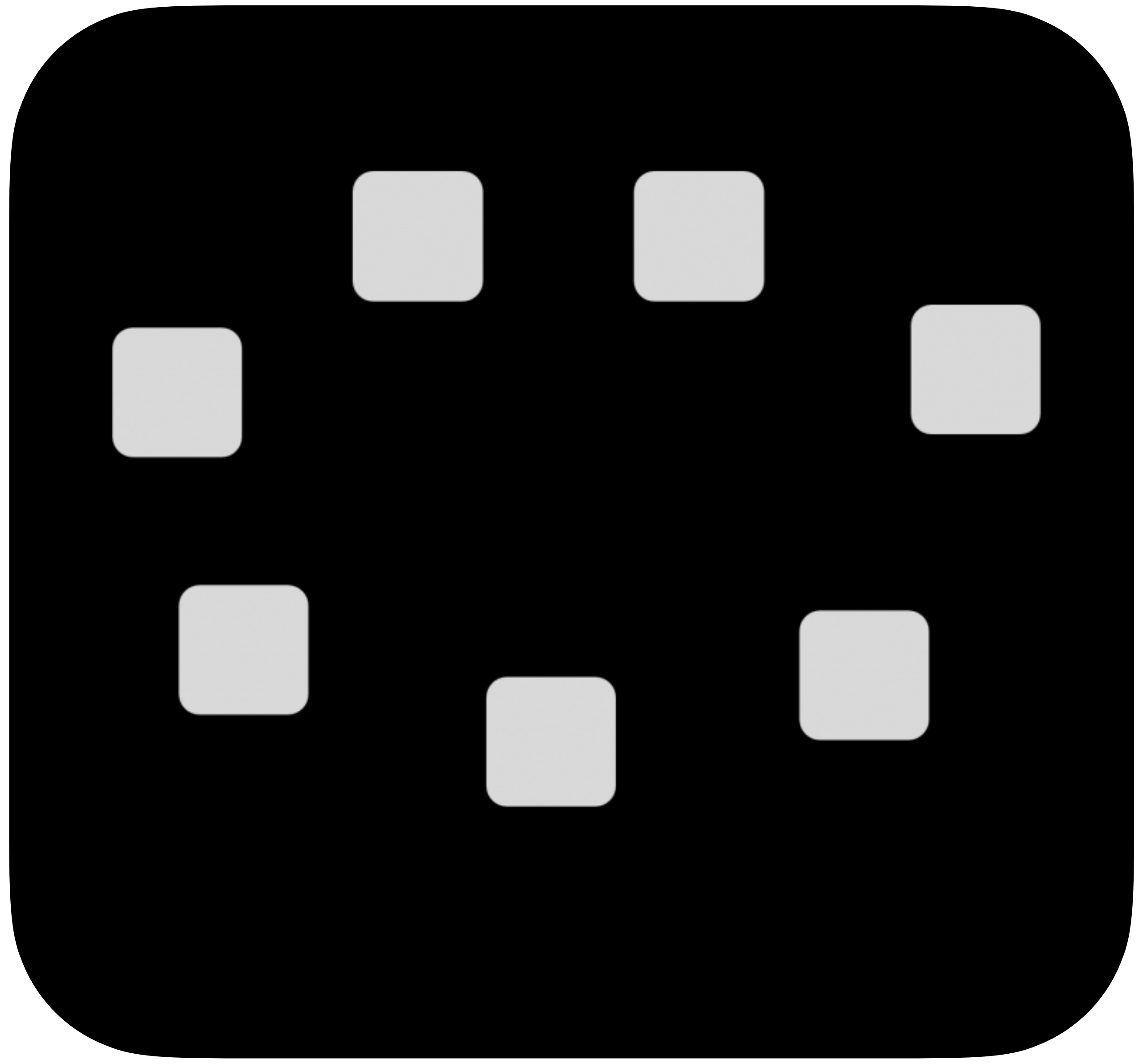
Designing for Error

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

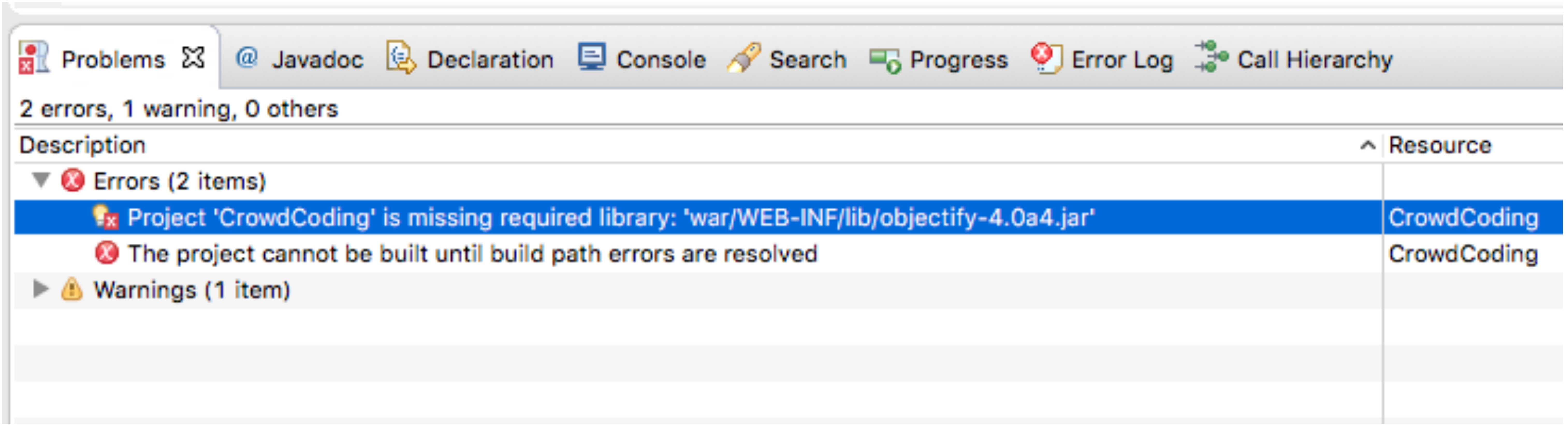
- Establishes relationship with user
- Important not to take user feel “stupid”
- Make the system take blame for errors
- Be positive, to encourage
- Provide helpful messages, not cute messages
- Avoid violent, negative, demeaning, threatening terms (e.g., illegal, invalid)

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)



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.

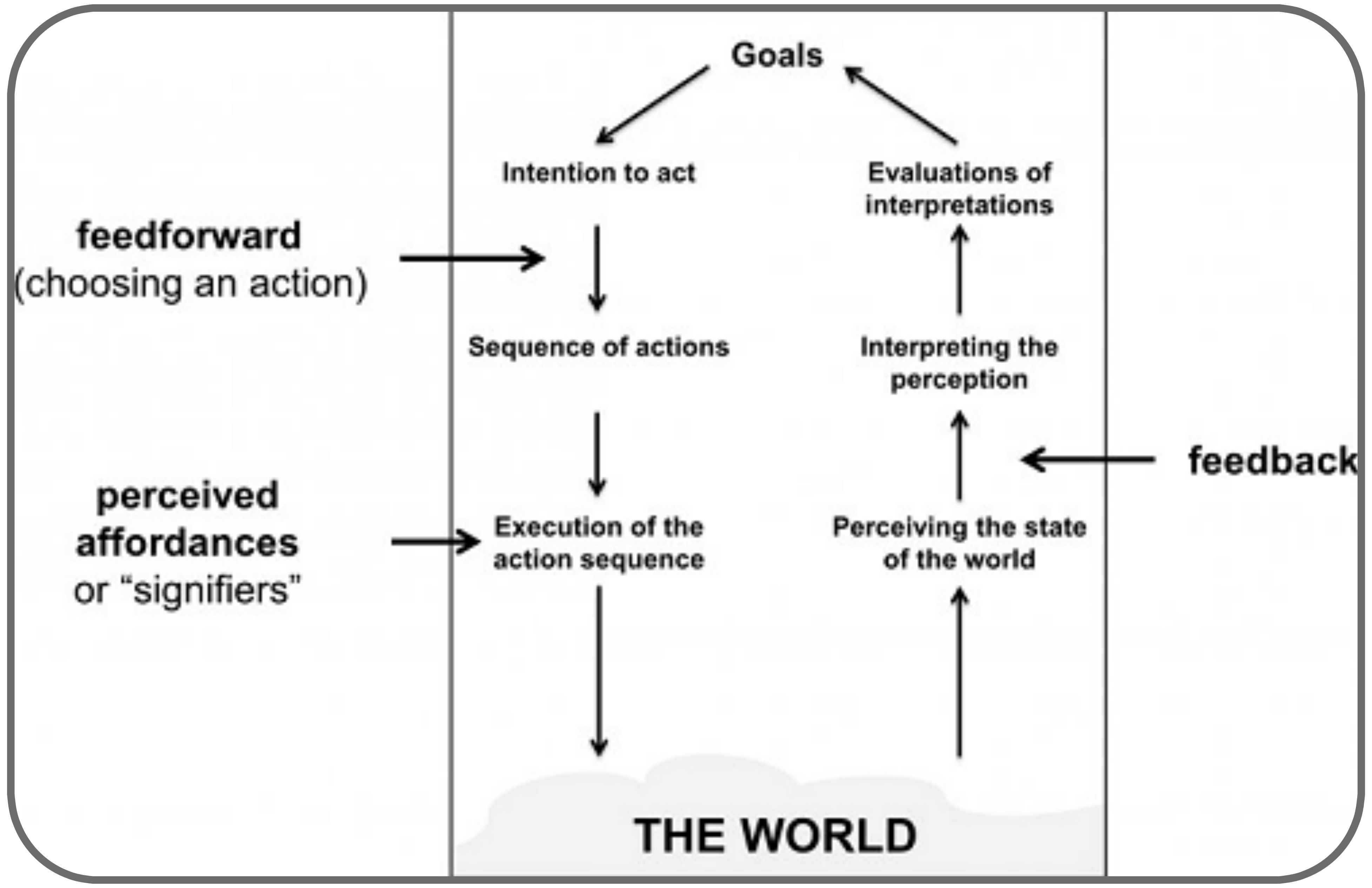
Norman’s Key Design Principles

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
    - <u>Execution:</u> Make options readily available
    - <u>Evaluation:</u> 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

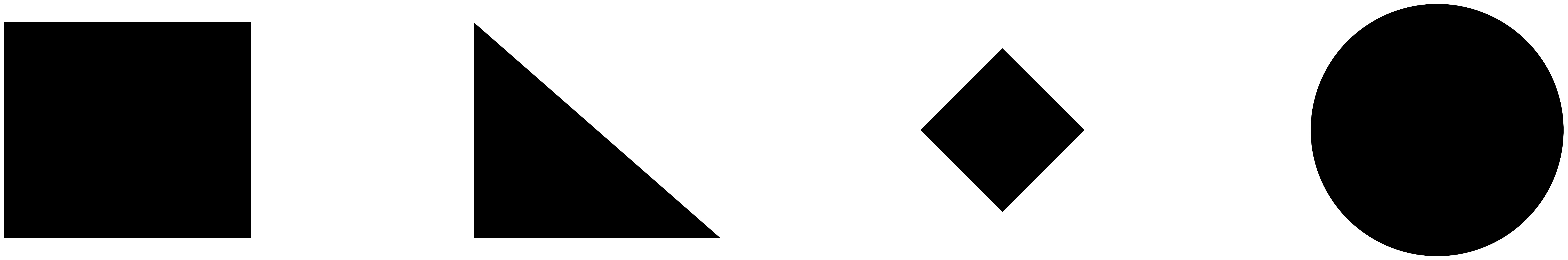


Key Questions

- 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 Characteristics

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

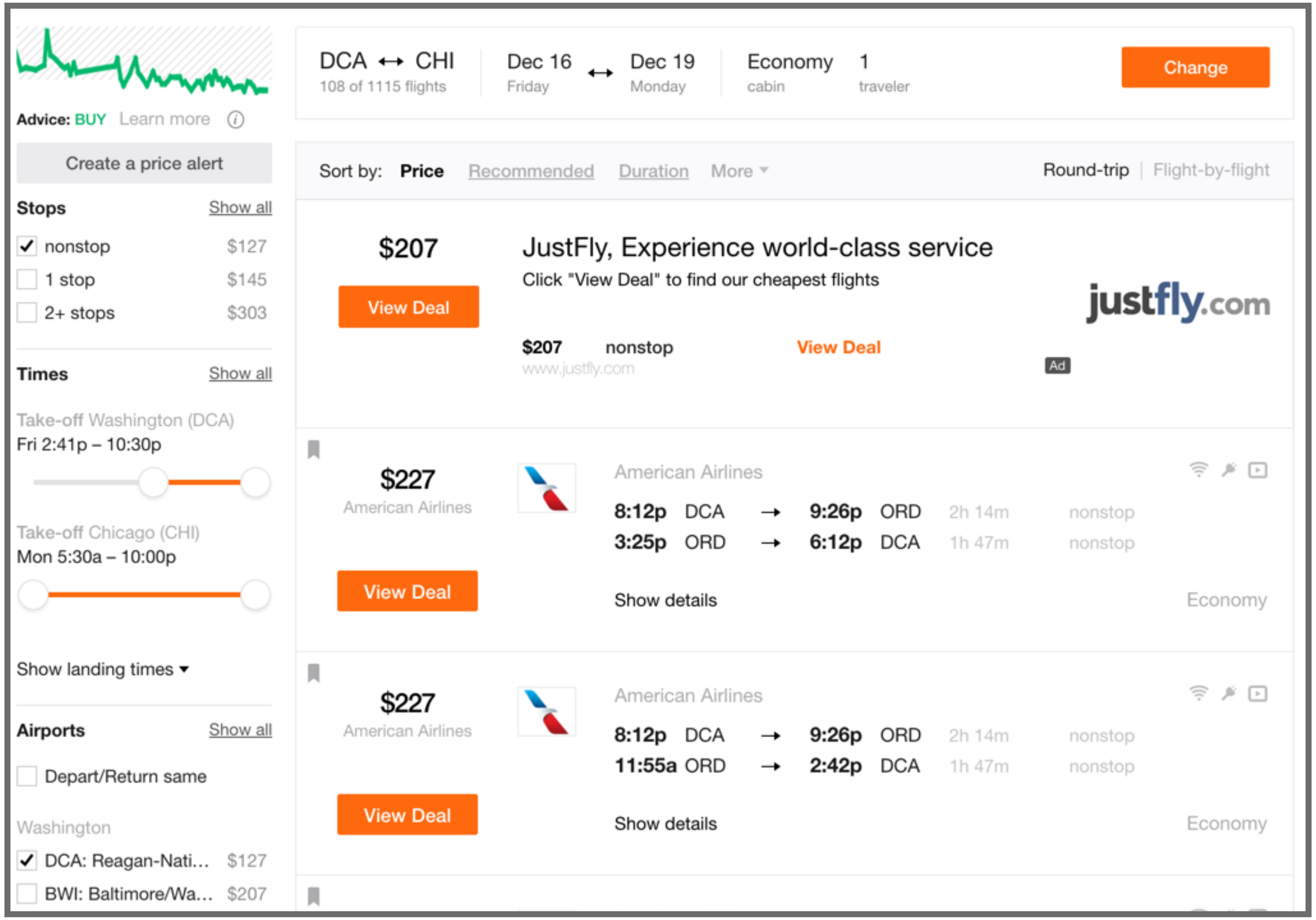
Benefits

- 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

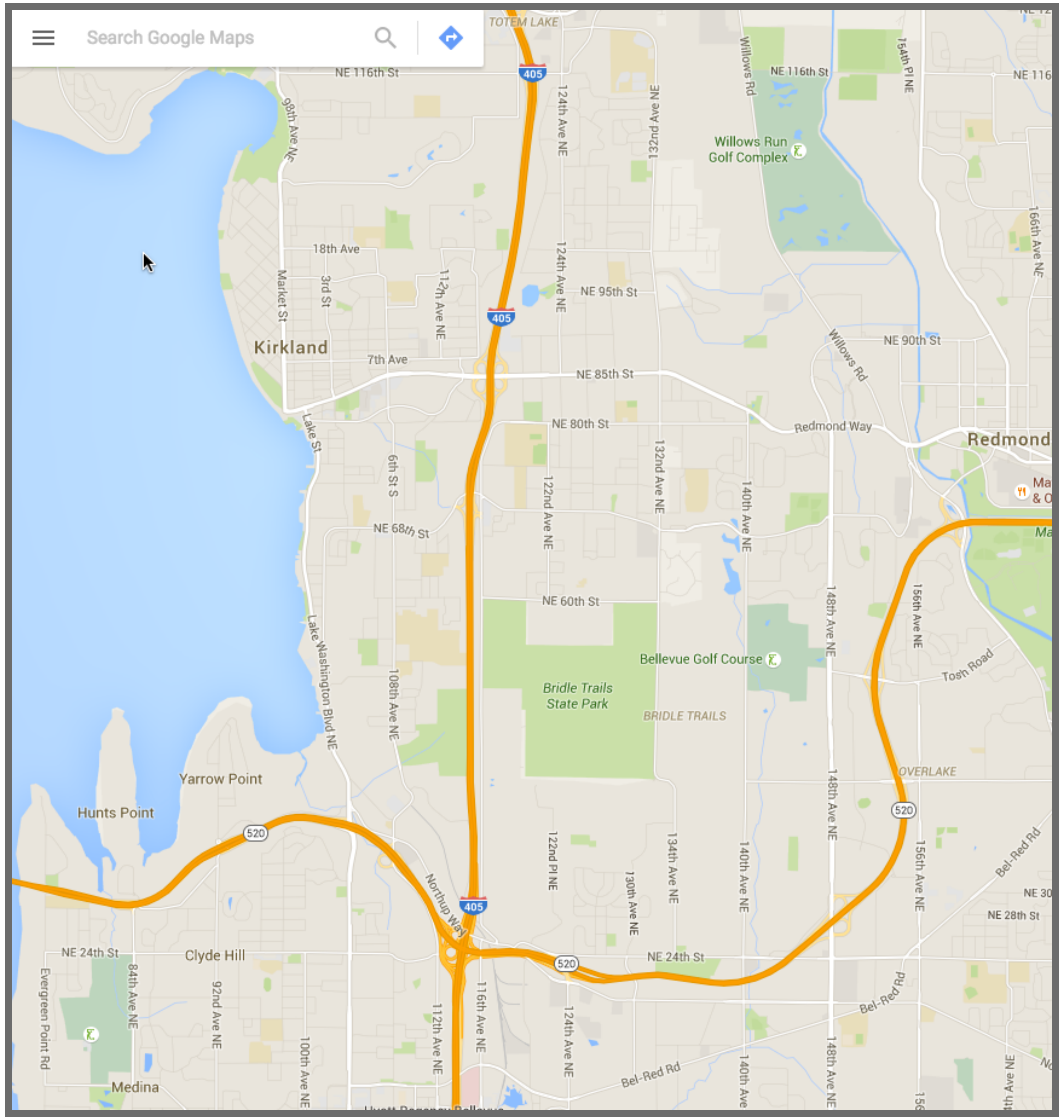
Drawbacks

- 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

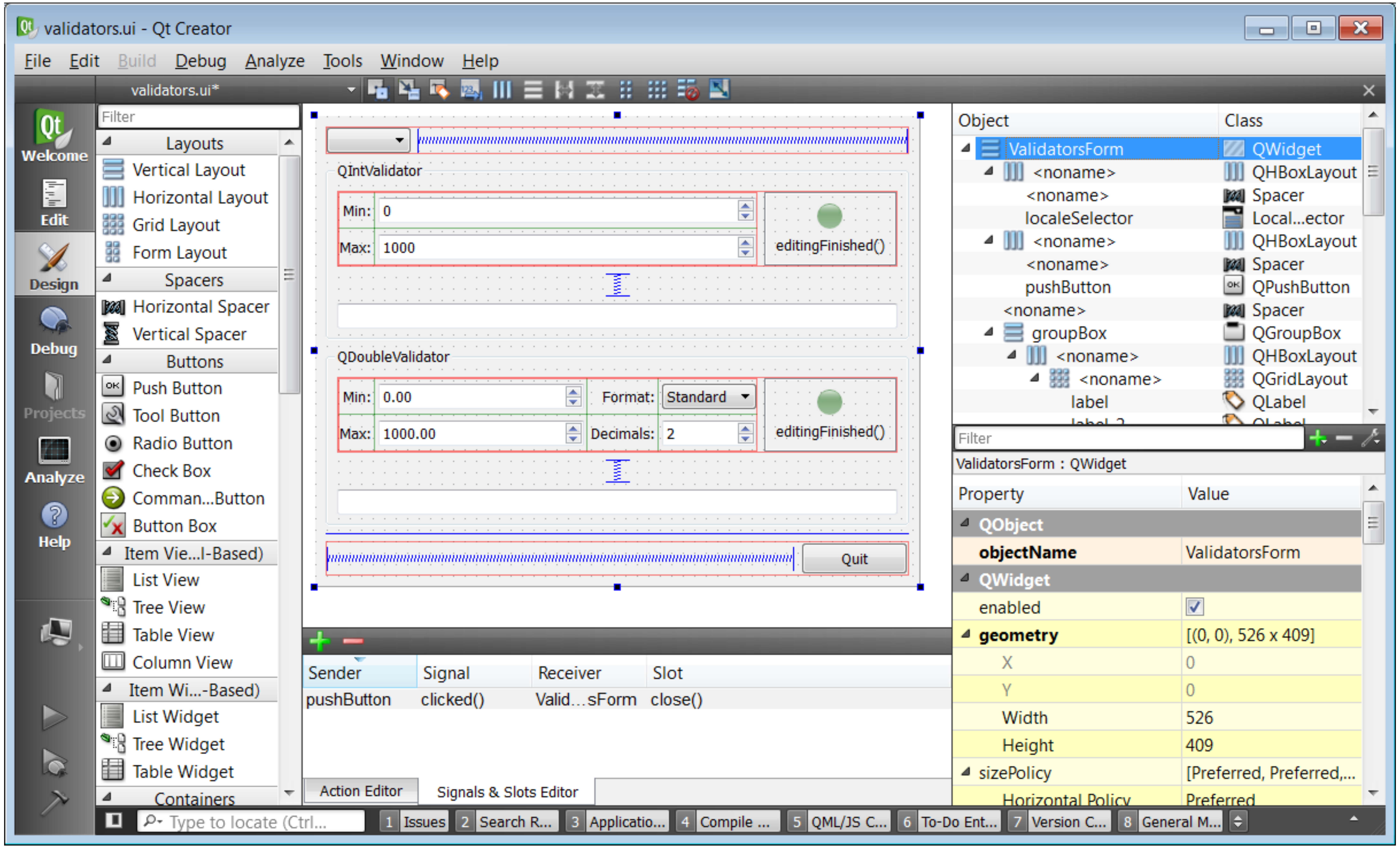
Example - Kayak



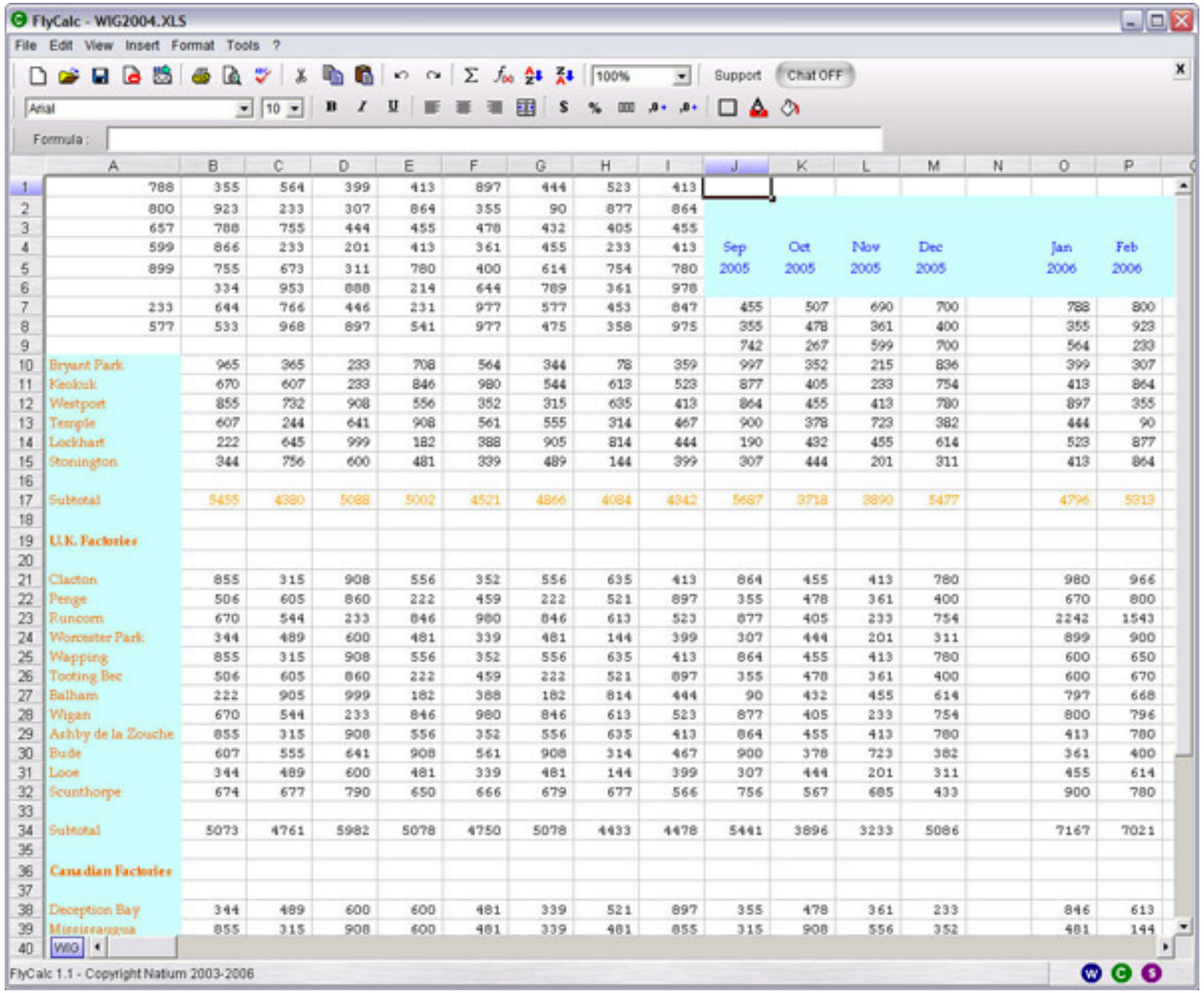
Example - Google Maps



Example - GUI Builder

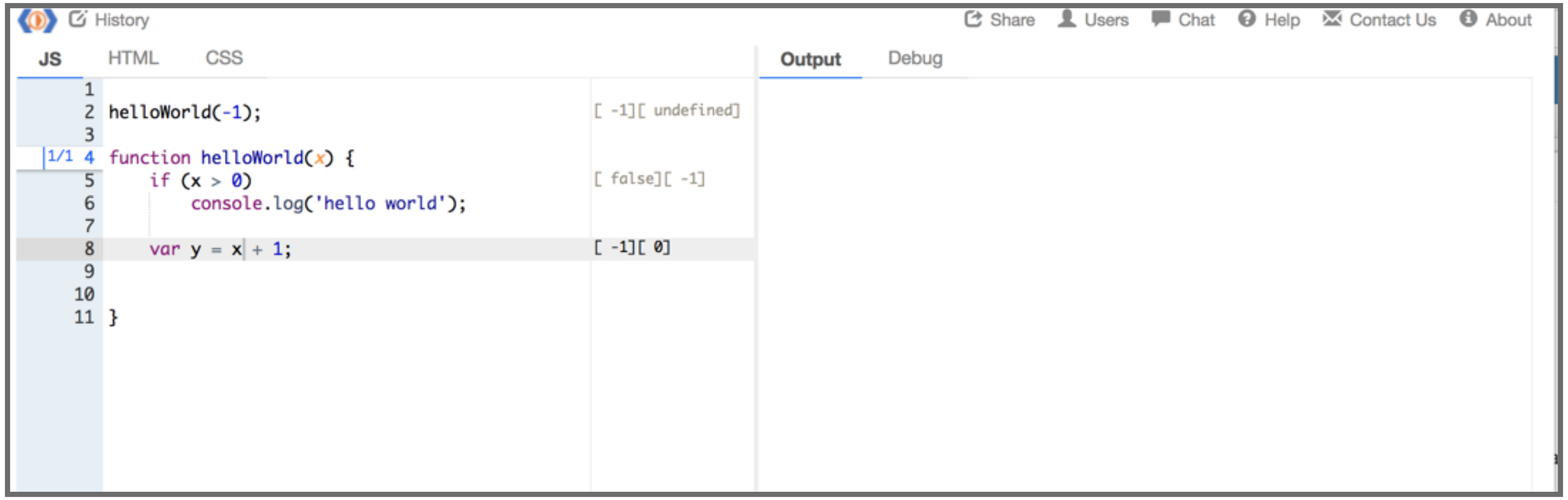


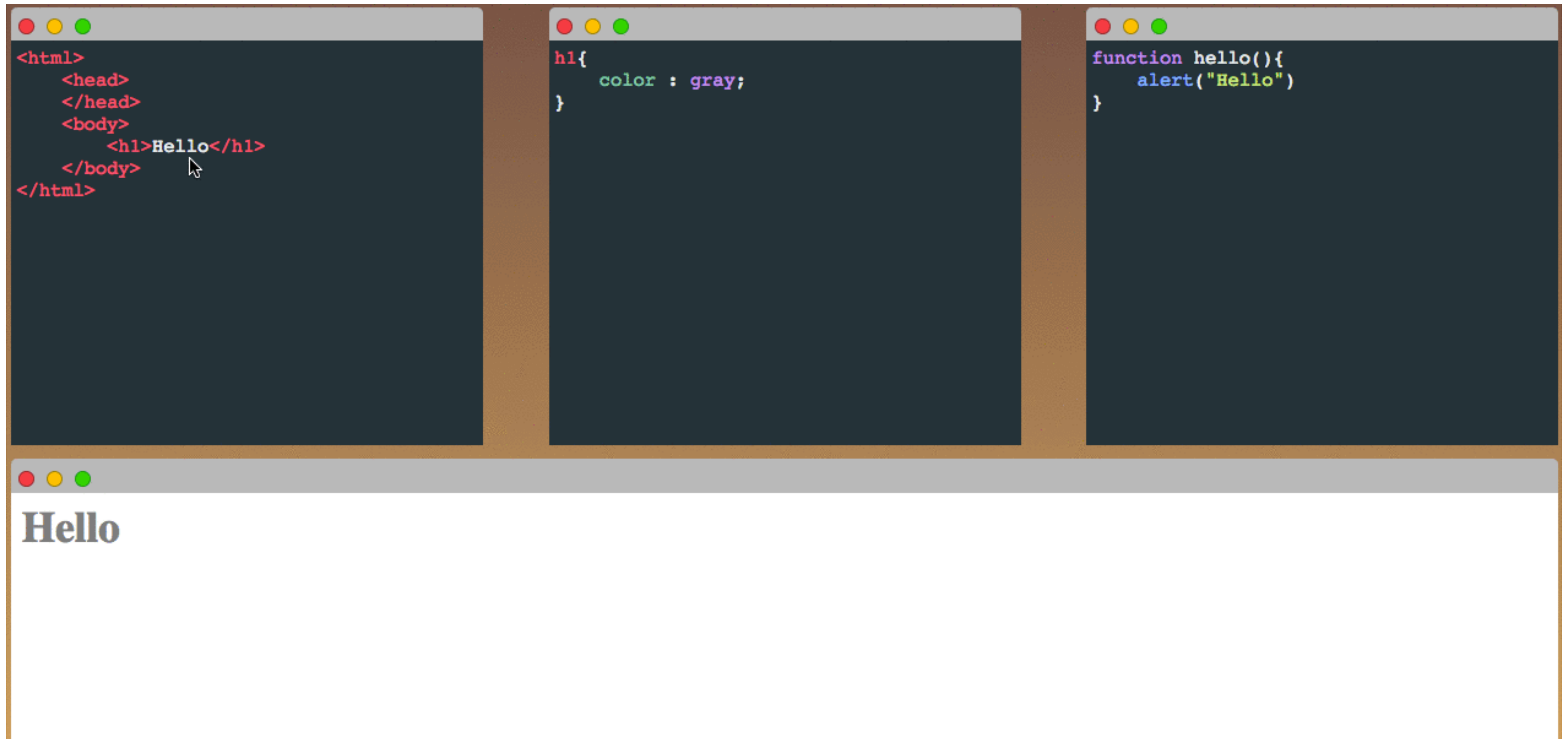
Example - Spreadsheets



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	788	355	564	399	413	897	444	523	413							
2	800	923	233	307	864	355	90	877	864							
3	657	788	755	444	455	478	432	405	455							
4	599	866	233	201	413	361	455	233	413	Sep	Oct	Nov	Dec		Jan	Feb
5	899	755	673	311	780	400	614	754	780	2005	2005	2005	2005		2006	2006
6		334	953	888	214	644	789	361	978							
7	233	644	766	446	231	977	577	453	847	455	507	690	700		788	800
8	577	533	968	897	541	977	475	358	975	355	478	361	400		355	923
9										742	267	599	700		564	233
10	Bryant Park	965	365	233	708	564	344	78	359	997	352	215	836		399	307
11	Keokuk	670	607	233	846	980	544	613	523	877	405	233	754		413	864
12	Westport	855	732	908	556	352	315	635	413	864	455	413	780		897	355
13	Temple	607	244	641	908	561	555	314	467	900	378	723	382		444	90
14	Lockhart	222	645	999	182	388	905	814	444	190	432	455	614		523	877
15	Stonington	344	756	600	481	339	489	144	399	307	444	201	311		413	864
16																
17	Subtotal	5455	4380	5088	5002	4521	4866	4084	4342	5687	3718	3890	5477		4796	5313
18																
19	U.K. Factories															
20																
21	Clacton	855	315	908	556	352	556	635	413	864	455	413	780		980	966
22	Penge	506	605	860	222	459	222	521	897	355	478	361	400		670	800
23	Runcorn	670	544	233	846	980	846	613	523	877	405	233	754		2242	1543
24	Worcester Park	344	489	600	481	339	481	144	399	307	444	201	311		899	900
25	Wapping	855	315	908	556	352	556	635	413	864	455	413	780		600	650
26	Tooting Bec	506	605	860	222	459	222	521	897	355	478	361	400		600	670
27	Balham	222	905	999	182	388	182	814	444	90	432	455	614		797	668
28	Wigan	670	544	233	846	980	846	613	523	877	405	233	754		800	796
29	Ashby de la Zouche	855	315	908	556	352	556	635	413	864	455	413	780		413	780
30	Bude	607	555	641	908	561	908	314	467	900	378	723	382		361	400
31	Looe	344	489	600	481	339	481	144	399	307	444	201	311		455	614
32	Scunthorpe	674	677	790	650	666	679	677	566	756	567	685	433		900	780
33																
34	Subtotal	5073	4761	5982	5078	4750	5078	4433	4478	5441	3896	3233	5086		7167	7021
35																
36	Canadian Factories															
37																
38	Deception Bay	344	489	600	600	481	339	521	897	355	478	361	233		846	613
39	Mississauga	855	315	908	600	481	339	481	855	315	908	556	352		481	144
40																

Example: Live Programming





10 Minute Break

In-Class Activity

In Class Activity: Direct Manipulation Programming

No Code Programming Environments

- In groups of 2 or 3
- Design a system for writing code through direct manipulation
    - Pick an application domain where your system will apply (e.g., banking, shipping, Android games)
    - 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 with annotations explaining application behavior
Due by 6:25pm