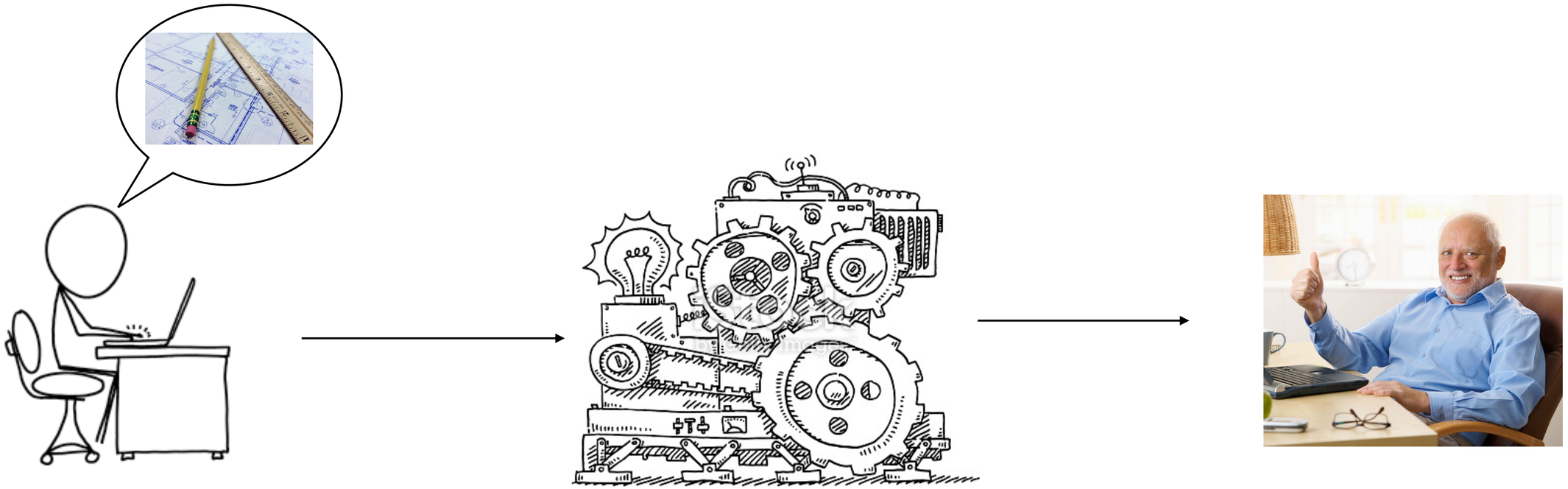
Interaction Modalities in Program Synthesis

Michael B. James

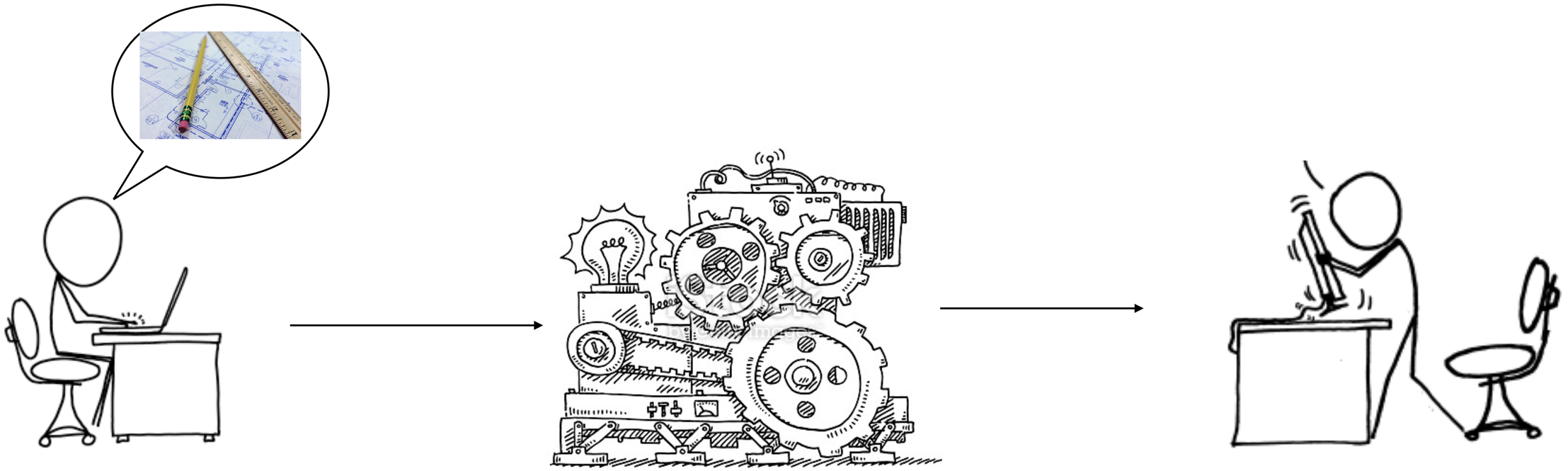
What is Program Synthesis?

Idea



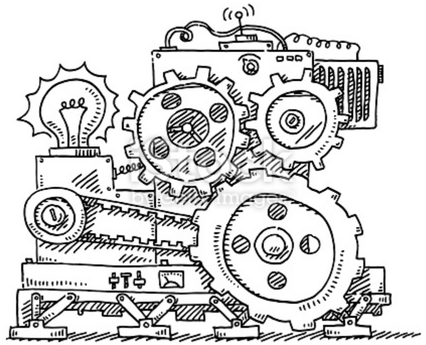
What is Program Synthesis?

Reality

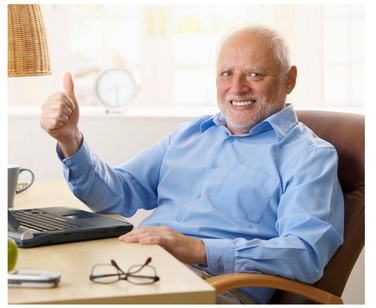


What is *Interactive* Program Synthesis?

Input



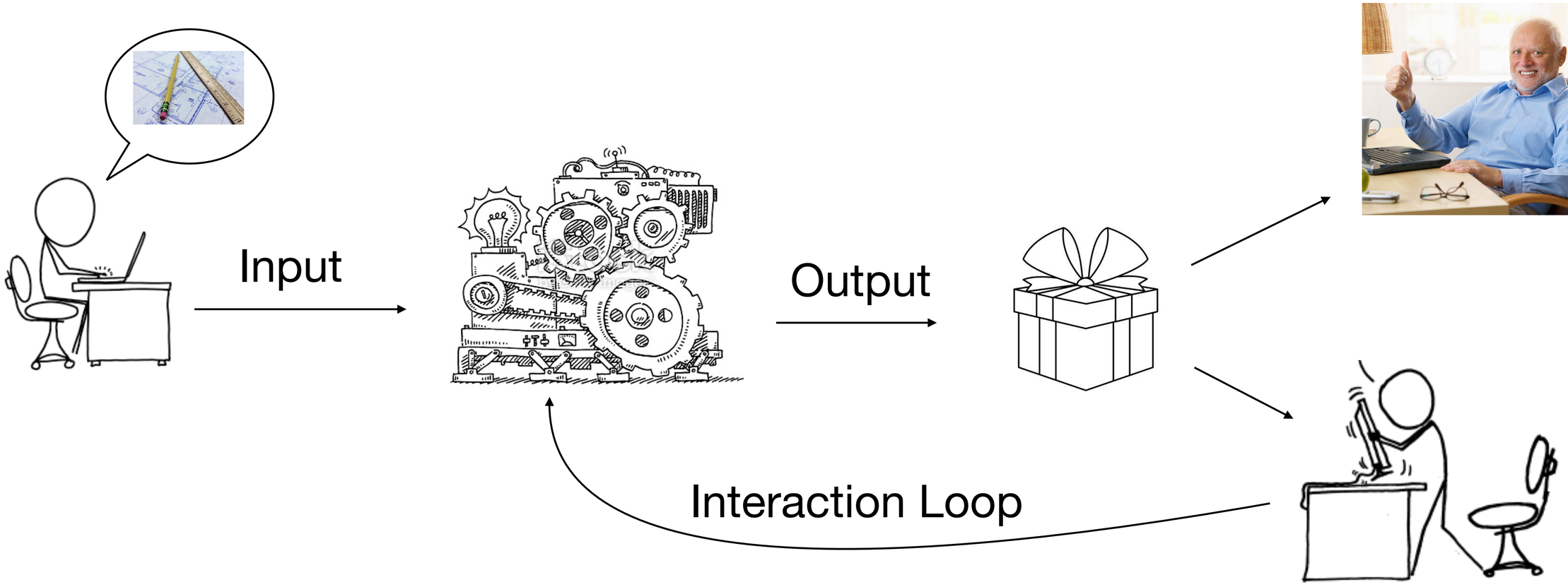
Output

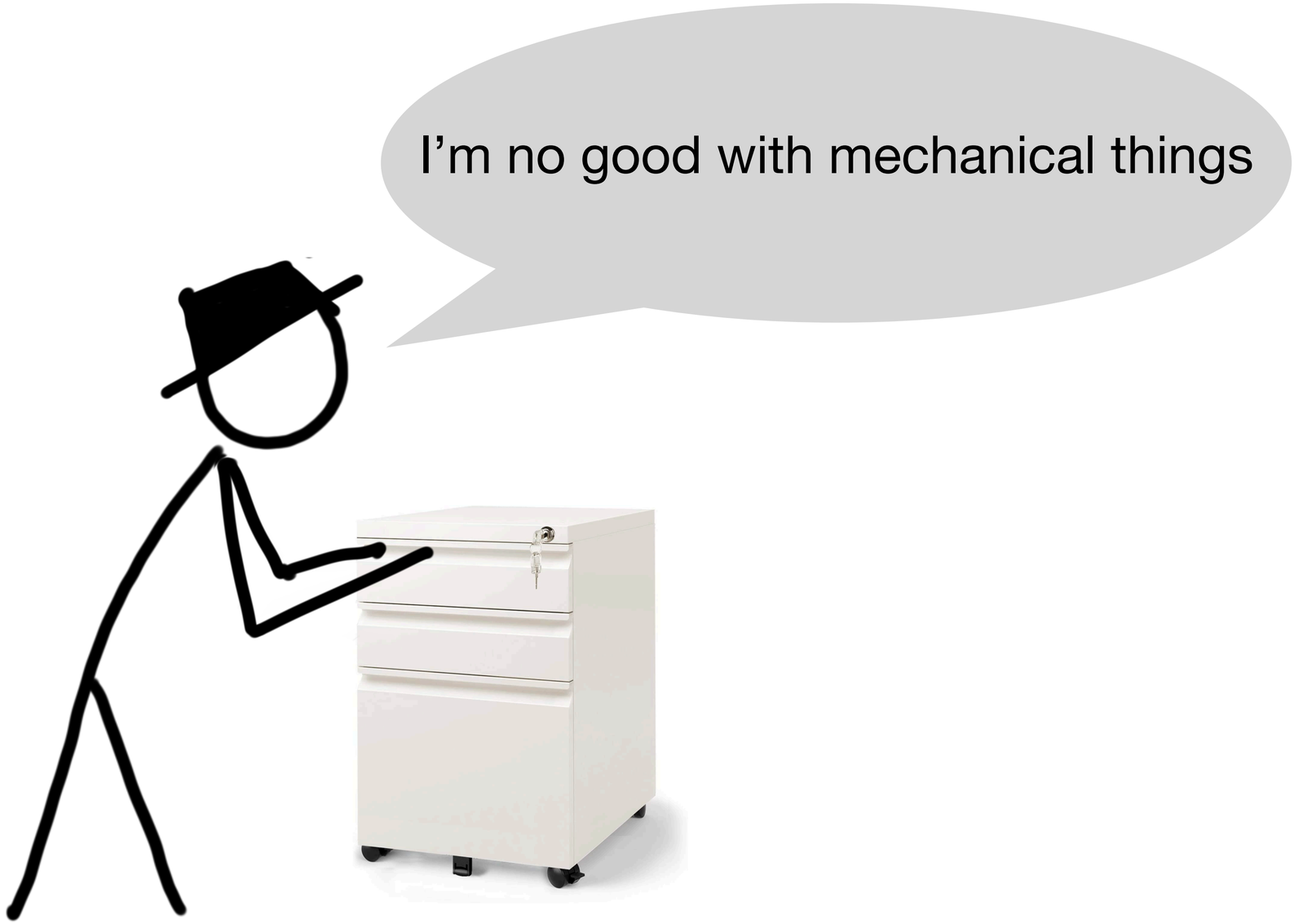


Interaction Loop

What is *Interactive* Program Synthesis?

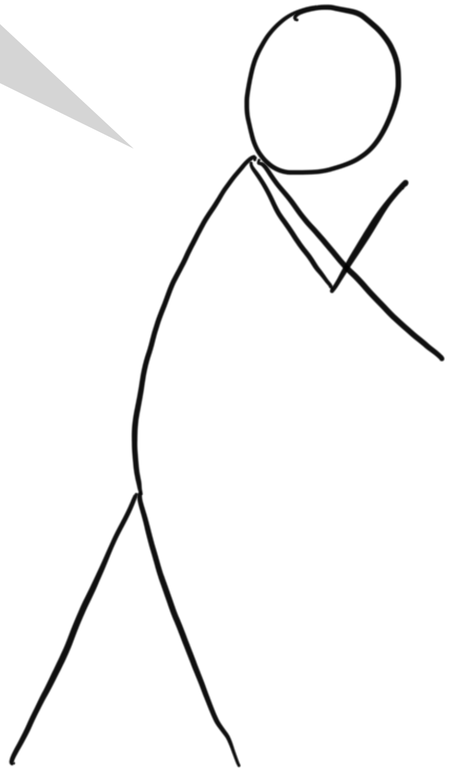
In the context of interaction theory

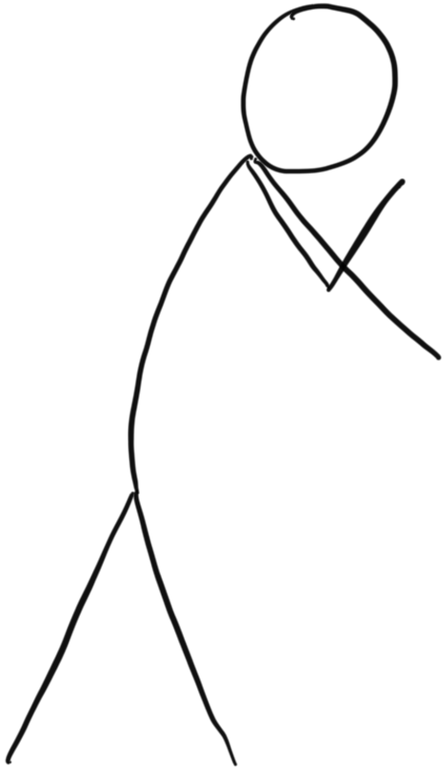


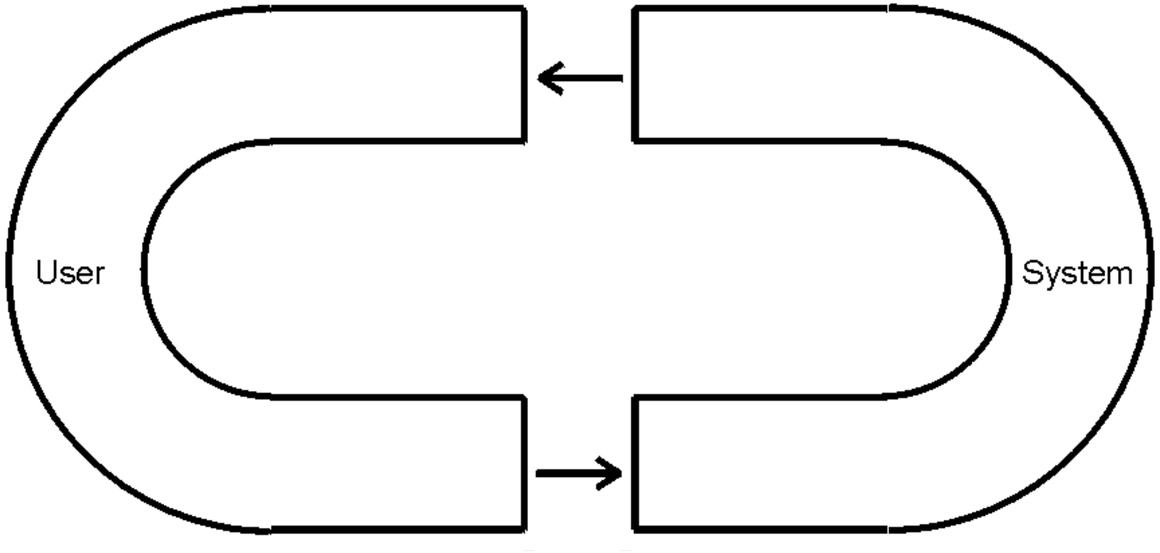

I’m no good with mechanical things

I think I understand how this works

Oh this did something







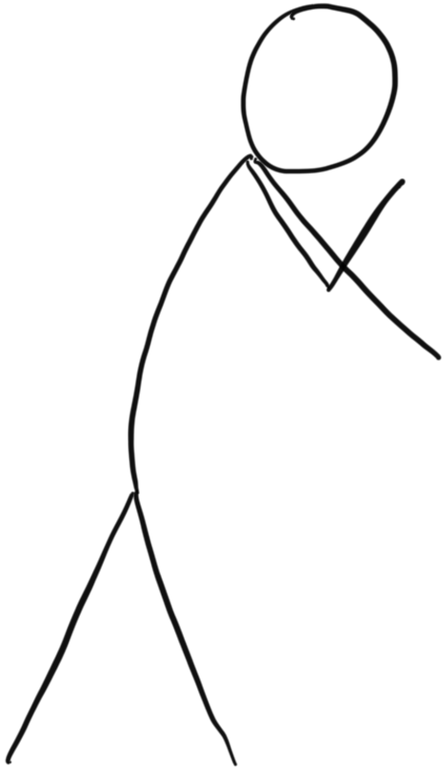
Articulation: see handles, specify intent

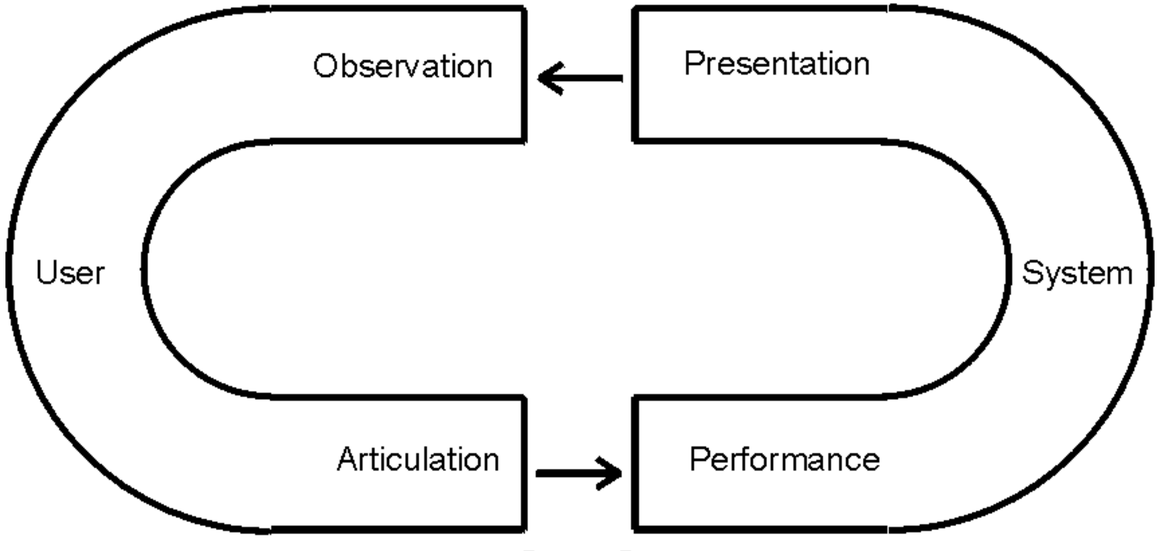
Performance: drawer tries to move, but doesn’t

Presentation: drawer doesn’t budge

Observation: did I achieve my goal?

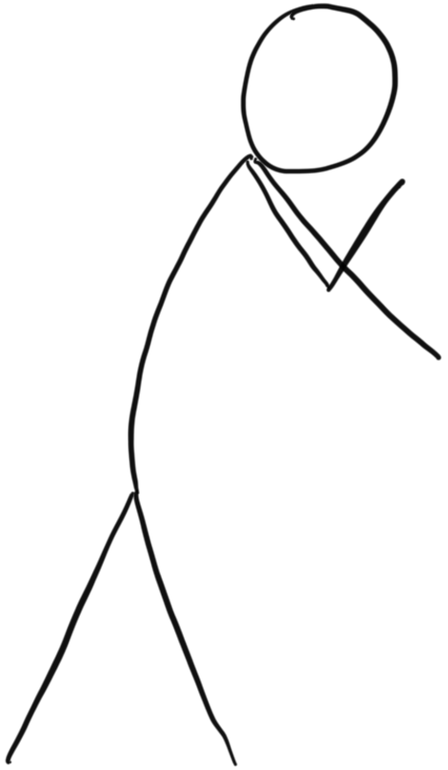
http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

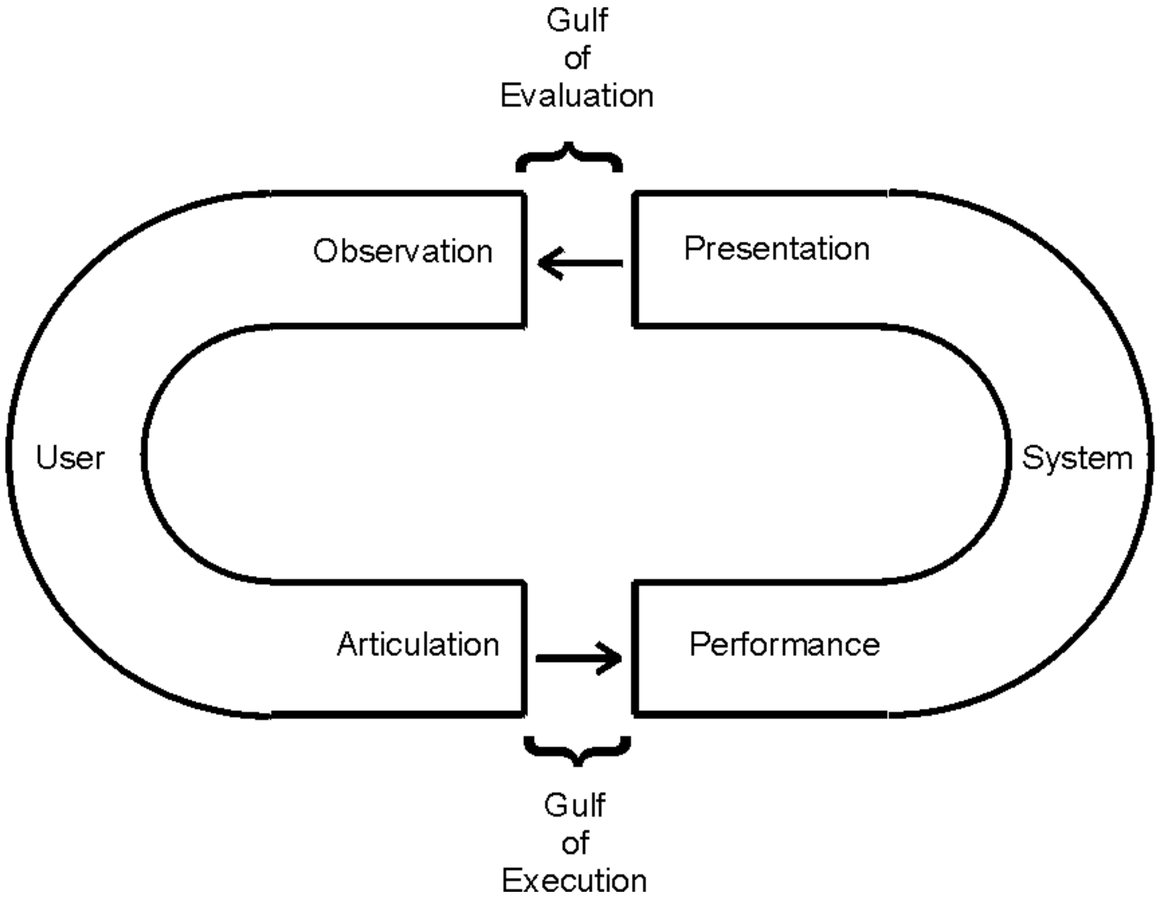




Gulf of Execution: Span between **expressing your intent** and the system **receiving that intent**

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

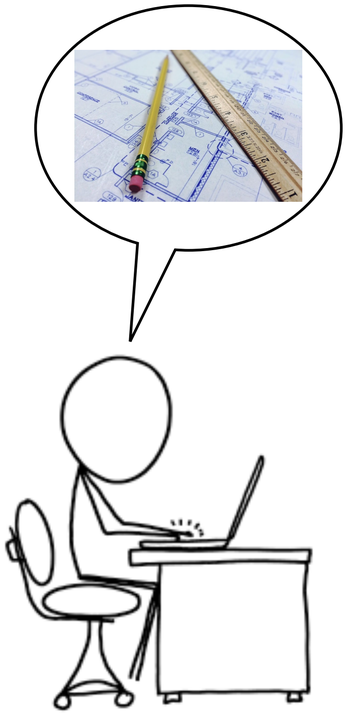




Gulf of Execution: Span between **expressing your intent** and the system **receiving that intent**

Gulf of Evaluation: Gap between the **language of the system** and the **language of your goal**

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/



Gulf of Evaluation

2. Output Styles

Observation

Presentation

User

3. Interaction Loop

System

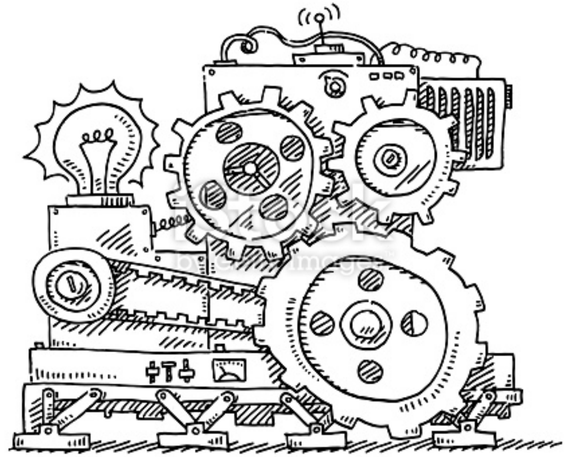
Articulation

Performance

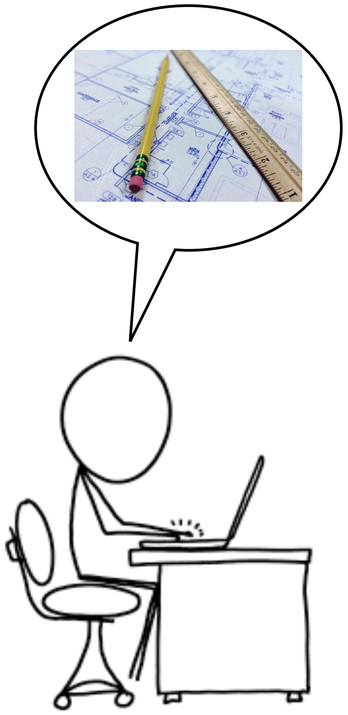
1. Input Styles

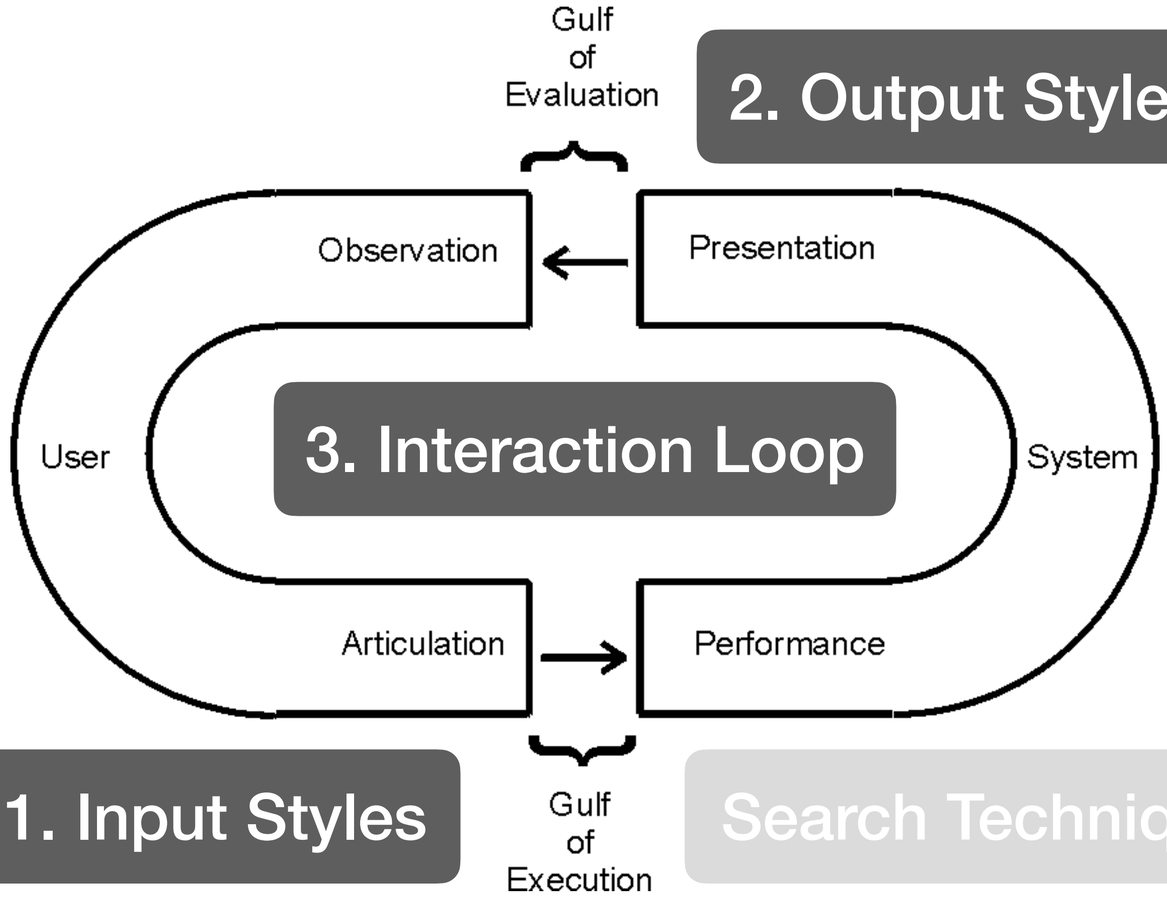
Gulf of Execution

Search Techniques



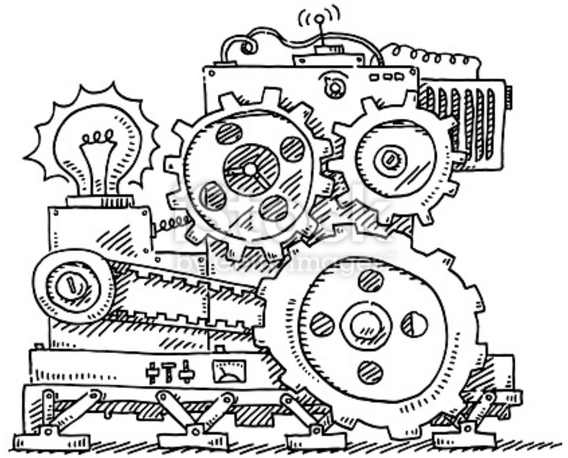
http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/





2. Output Styles

3. Interaction Loop



1. Input Styles

Search Techniques

4. My Work

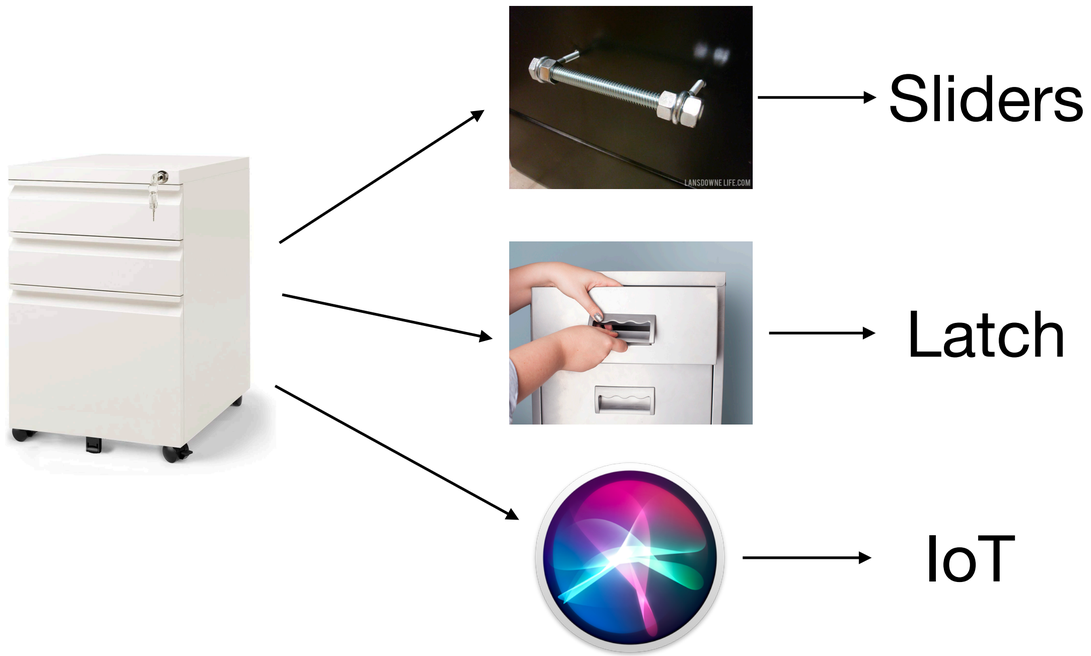
Hoogle+

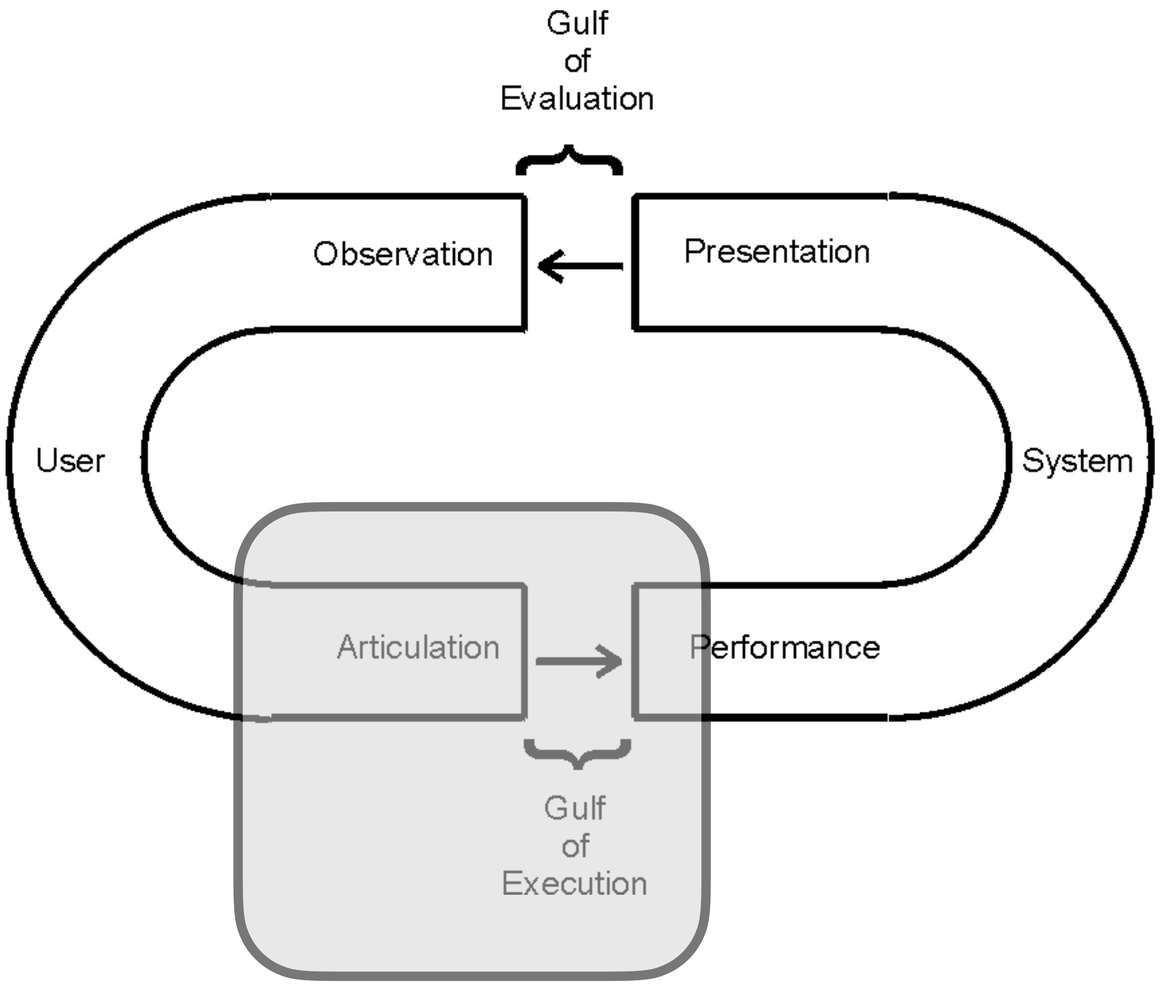
5. Open Questions

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

Input Styles

Input Styles



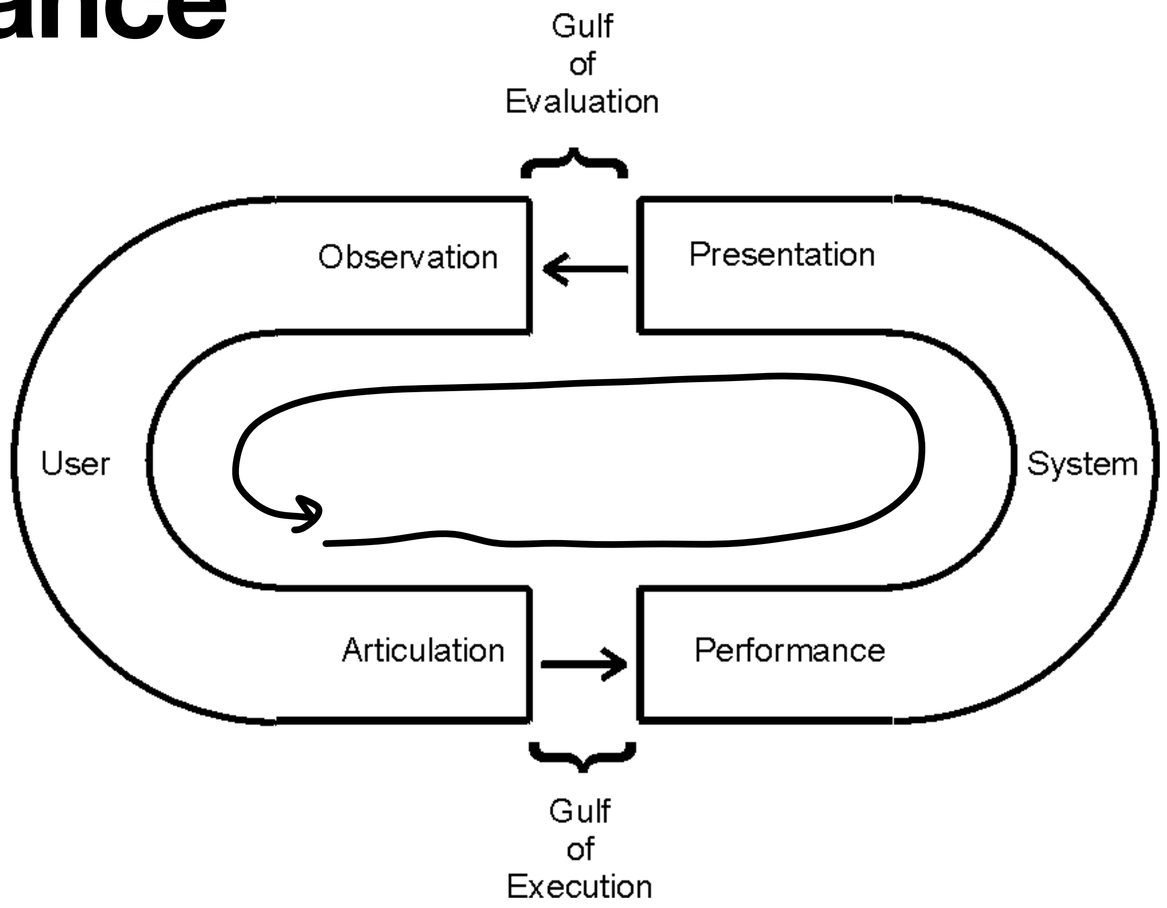


Input style dictates search performance

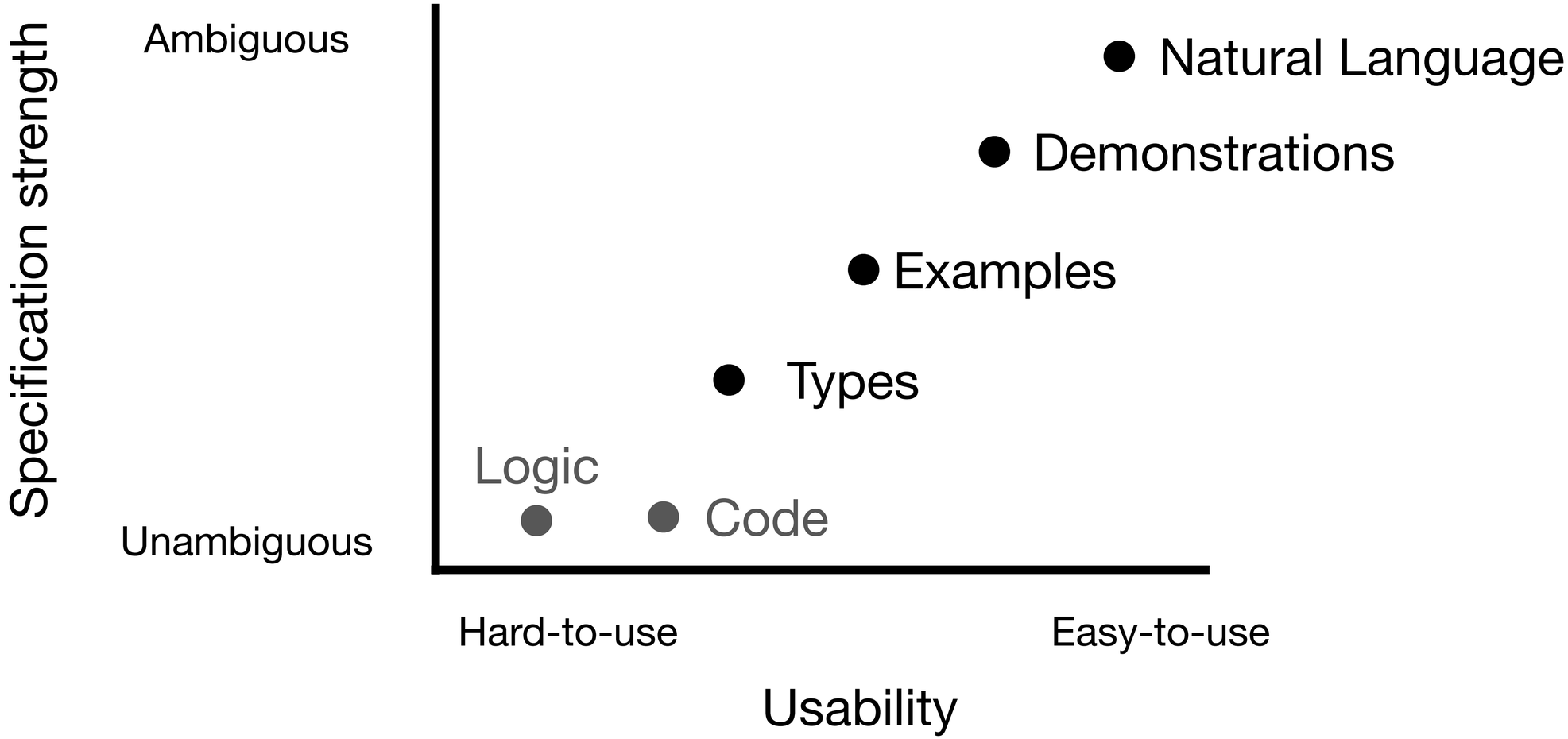
Input Styles vs Performance

Input Style: Usability

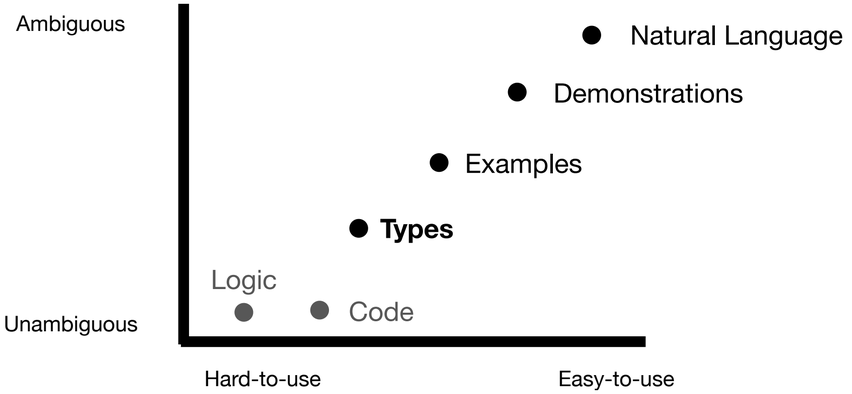
Performance: Ambiguity

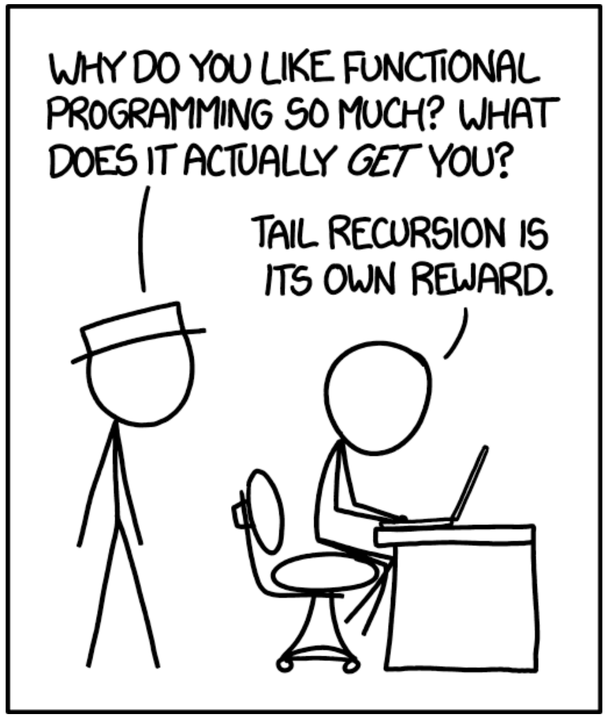


Input Styles vs Performance



Input Styles - Types



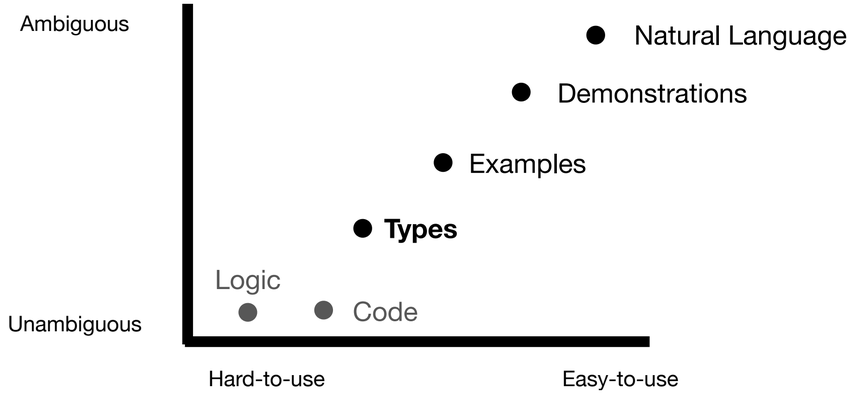


Types Only Club

Types & Examples

Input Styles - Types

Types Only Club



- Specific enough to articulate intent

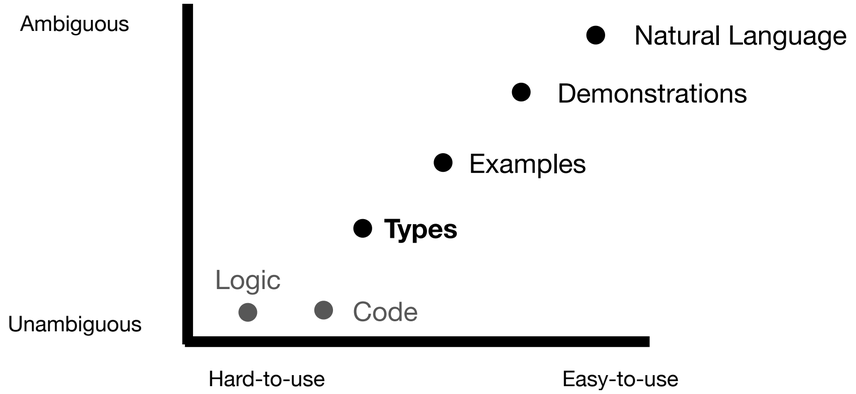
What sort of types?

- Support effective search

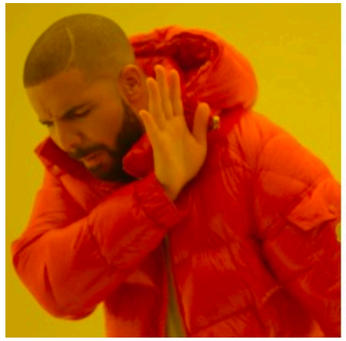
[1]
N. Polikarpova, I. Kuraj, and A. Solar-Lezama, “Program Synthesis from Polymorphic Refinement Types,” in PLDI 2016

Input Styles - Types

Types Only Club



What sort of types?

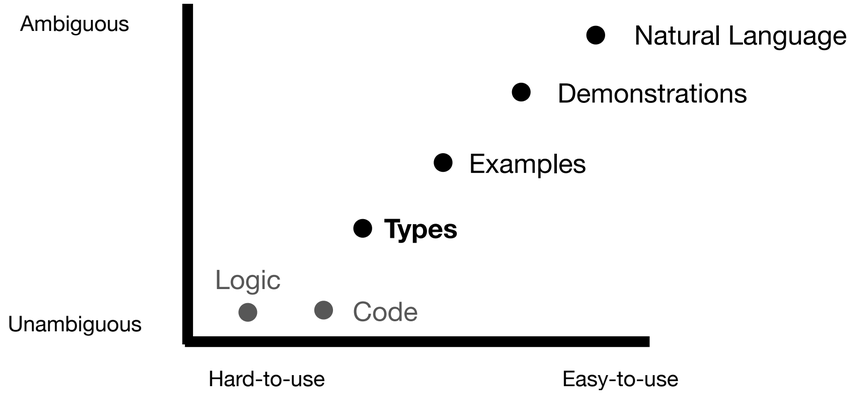


Haskell Types

[1] N. Polikarpova, I. Kuraj, and A. Solar-Lezama, “Program Synthesis from Polymorphic Refinement Types,” in PLDI 2016

Input Styles - Types

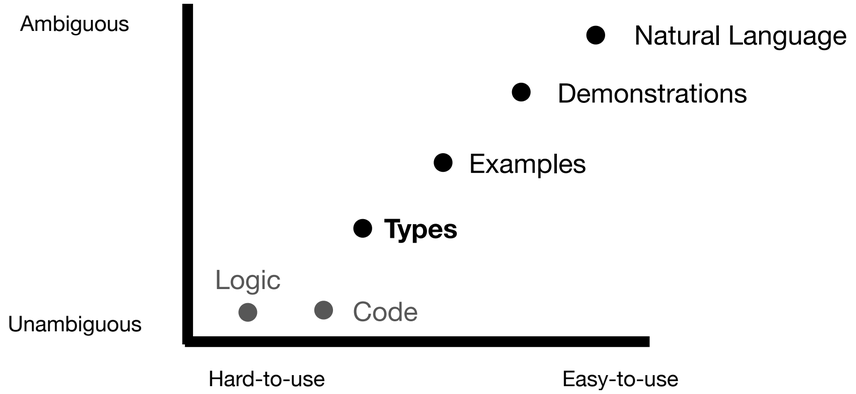
Types Only Club



What sort of types?

Input Styles - Types

Types Only Club



- Static Types +
  - Resource Refinements[2]
  - Value Refinements[1]
  - Privacy Refinements[3]
  - …

Task: stutter inputs twice

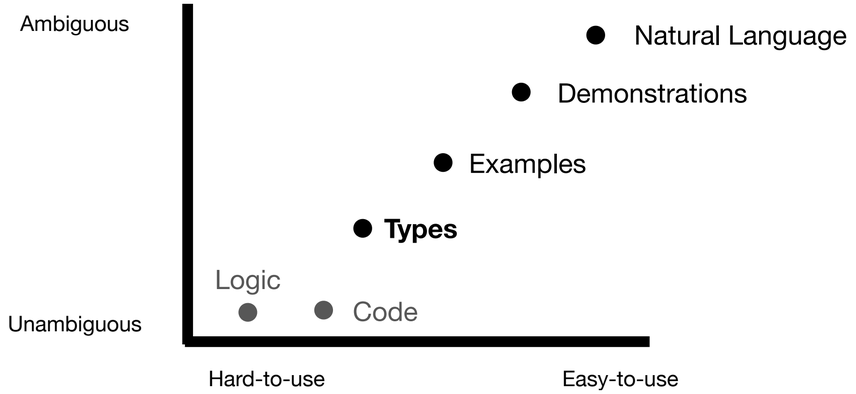
```
stutter “abc” == “aabbcc”
```

If you can typecheck it, you can synthesize it!

[1] N. Polikarpova, I. Kuraj, and A. Solar-Lezama, “Program Synthesis from Polymorphic Refinement Types,” in PLDI 2016
[2] T. Knoth, D. Wang, N. Polikarpova, and J. Hoffmann, “Resource-guided program synthesis,” in PLDI 2019.
[3] C. Smith and A. Albarghouthi, “Synthesizing Differentially Private Programs,” ICFP 2019,

Input Styles - Types

Types Only Club



Thinking this up can be hard

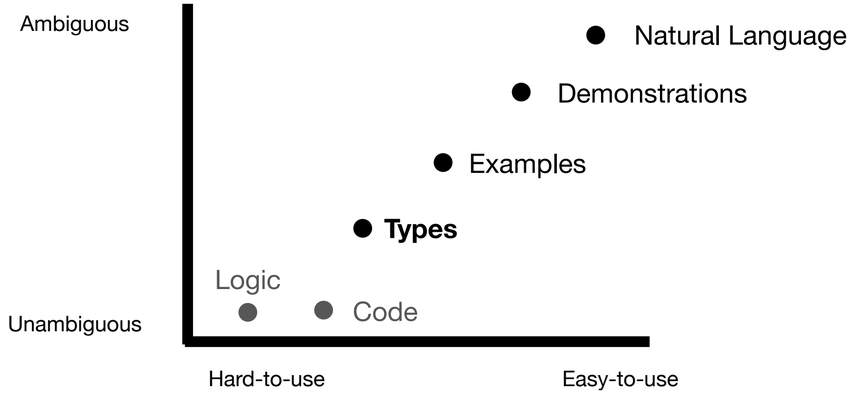

```
stutter :: xs: List a -> {List a | len _v == (len xs) * 2}
```

Task: stutter inputs twice

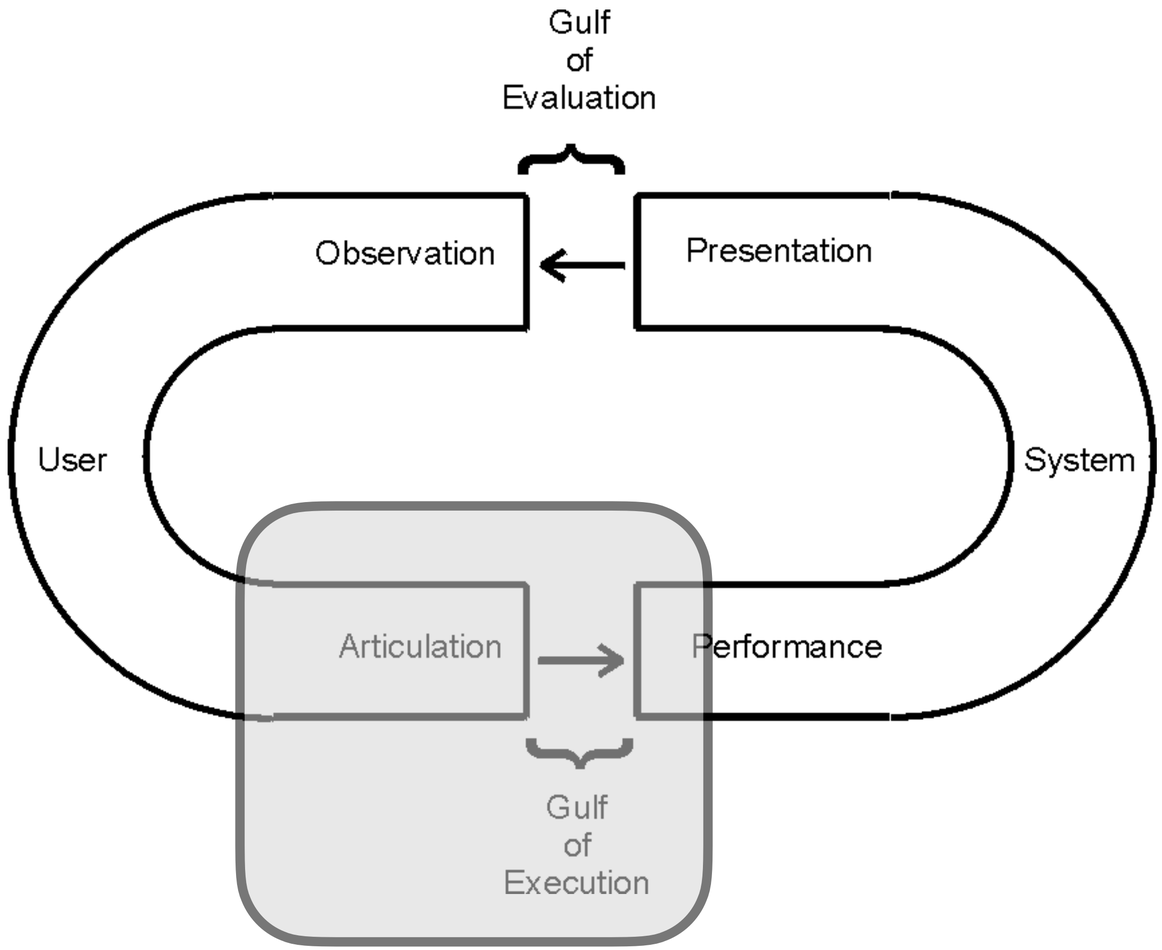
```
stutter “abc” == “aabbcc”
```

Input Styles - Types

Types + Example Club

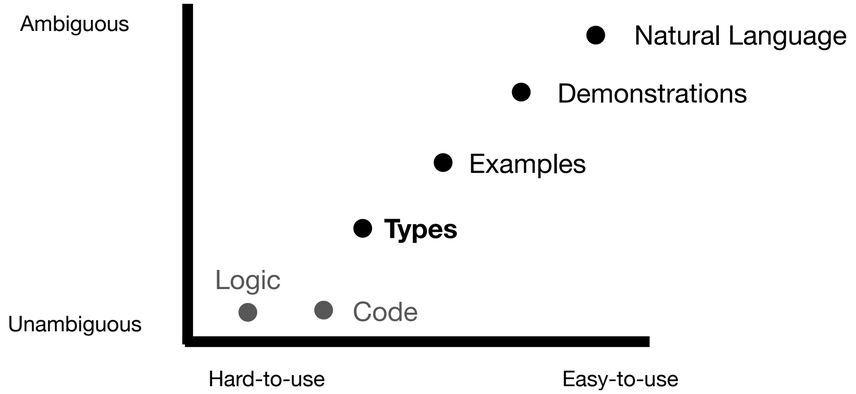


Trade refinement types
and
still cross the Gulf of Execution?



Input Styles - Types

Types + Example Club



Java Type Query

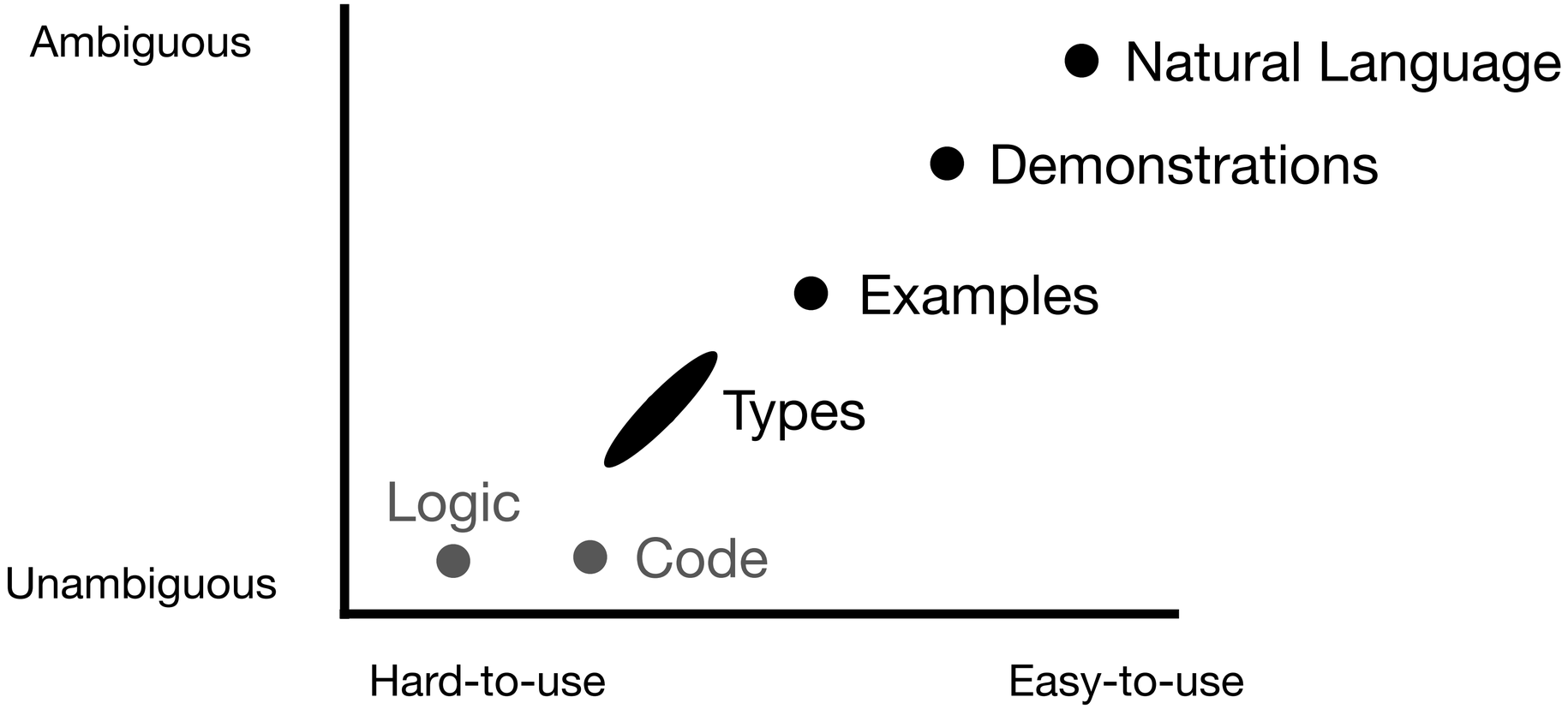
Input/Output Examples

Java Function

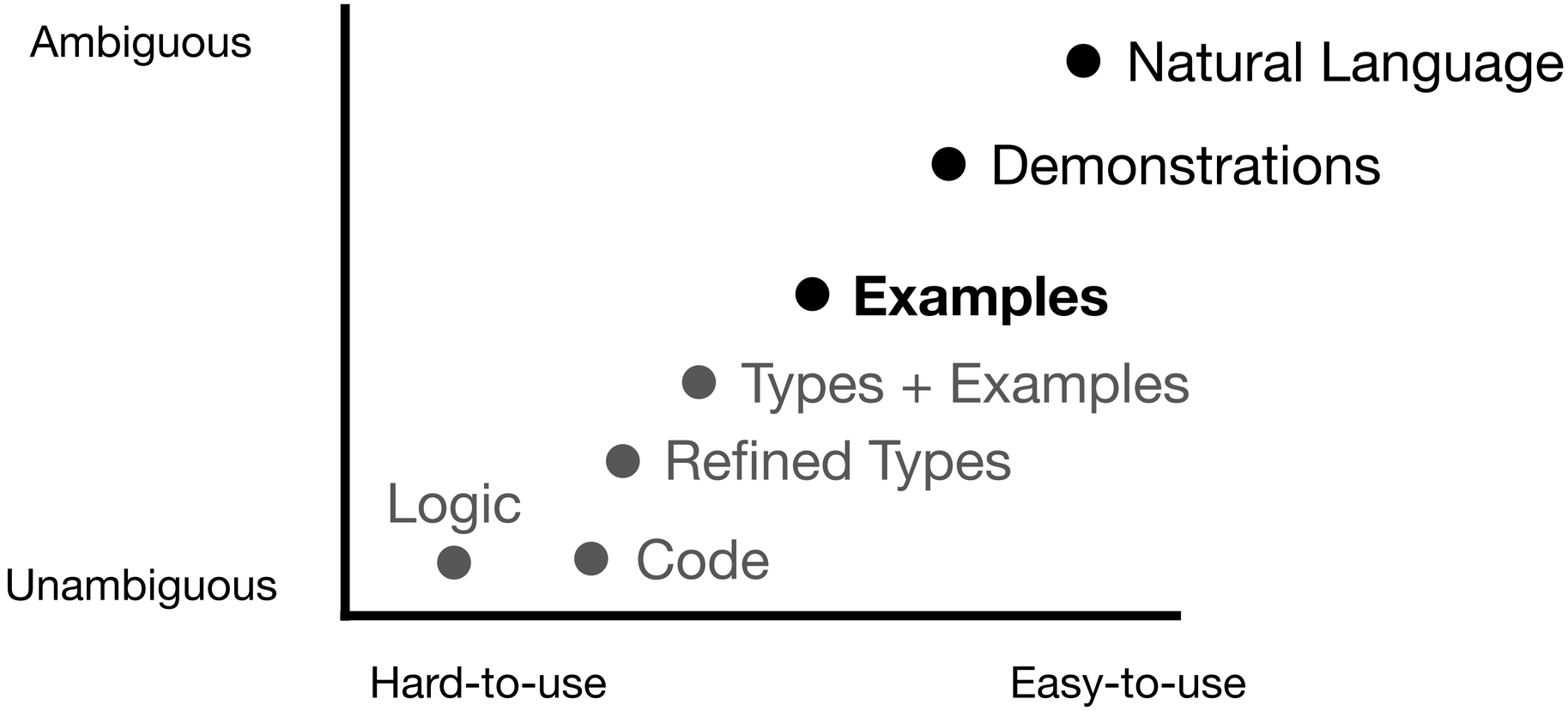
```
Area rotate(Area obj, Point2D pt, double angle) {
    AffineTransform at = new AffineTransform();
    double x = pt.getX();
    double y = pt.getY();
    at.setToRotation(angle, x, y);
    Area obj2 = obj.createTransformedArea(at);
    return obj2;
}
```

Feng et al. POPL 2017

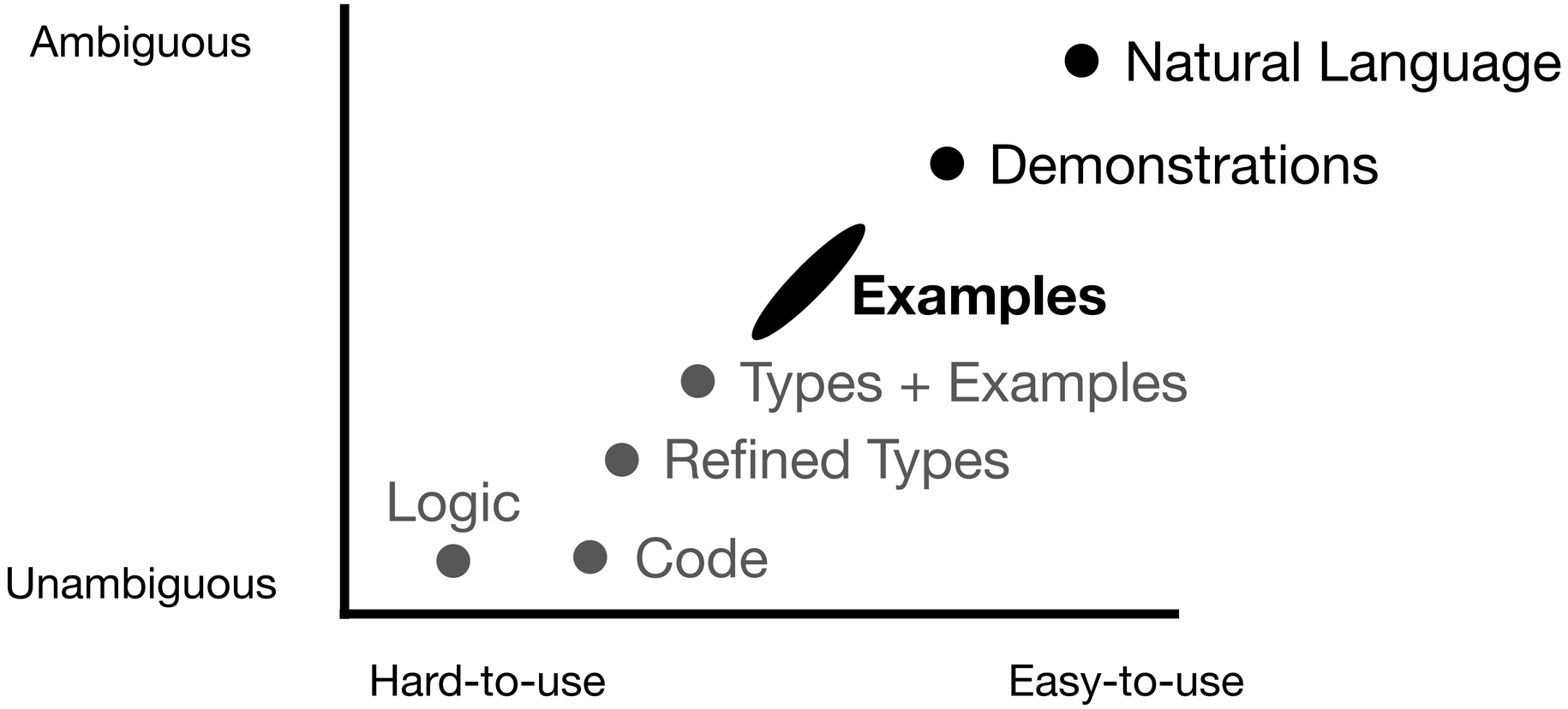
Input Styles



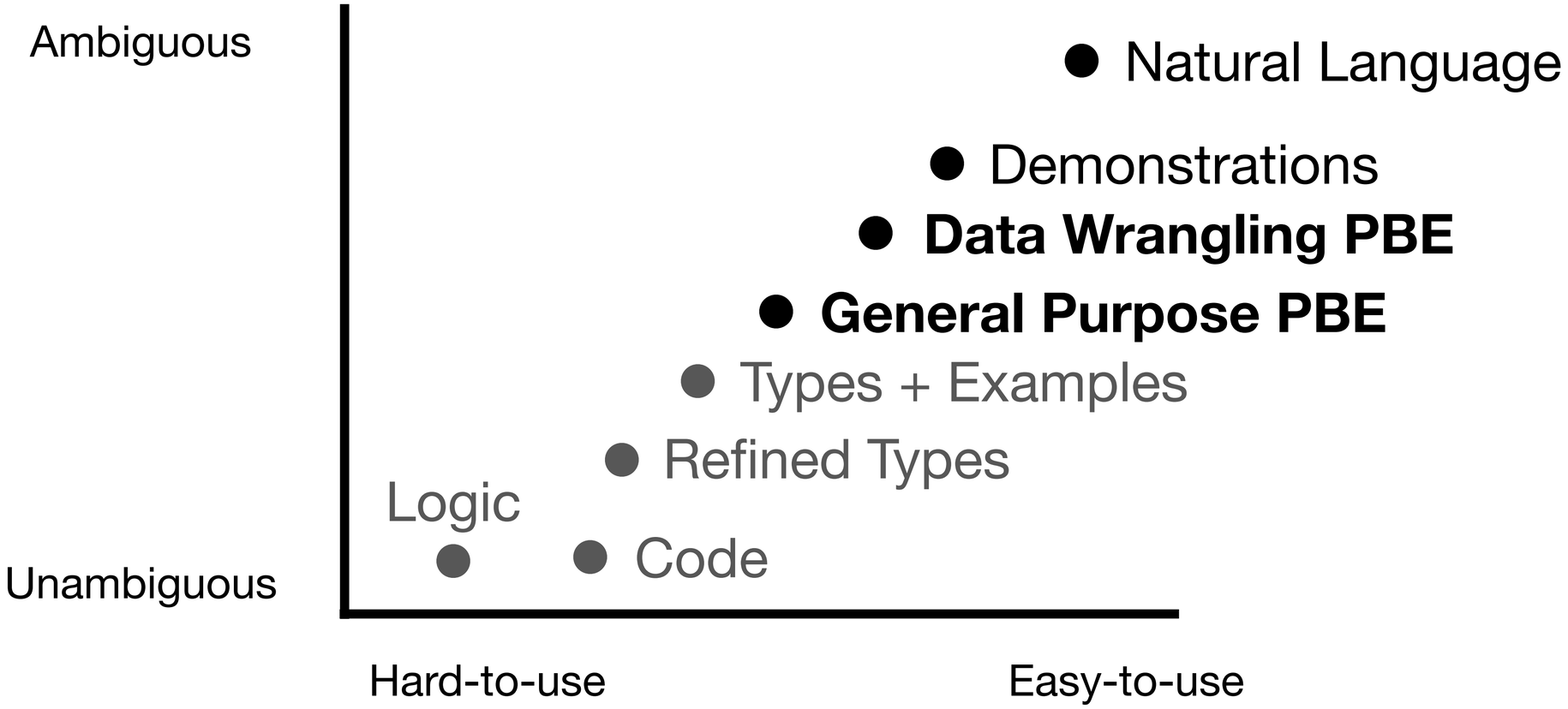
Input Styles - Examples



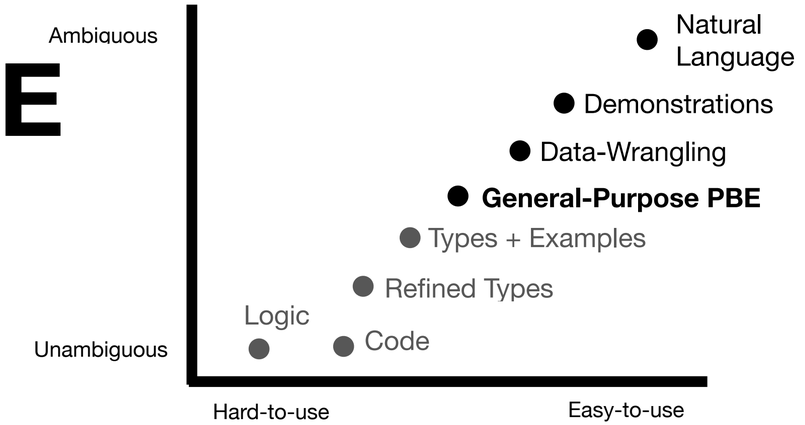
Input Styles - Examples



Input Styles - Examples

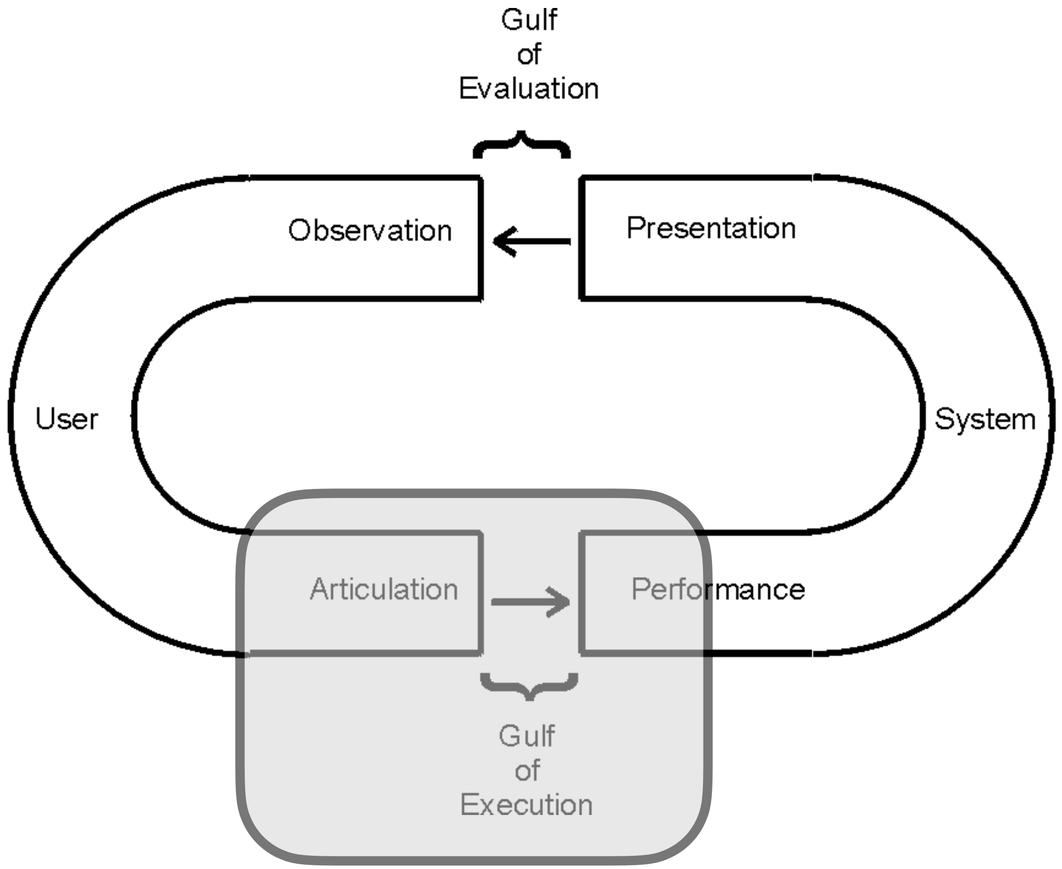


Input Styles - General Purpose PBE



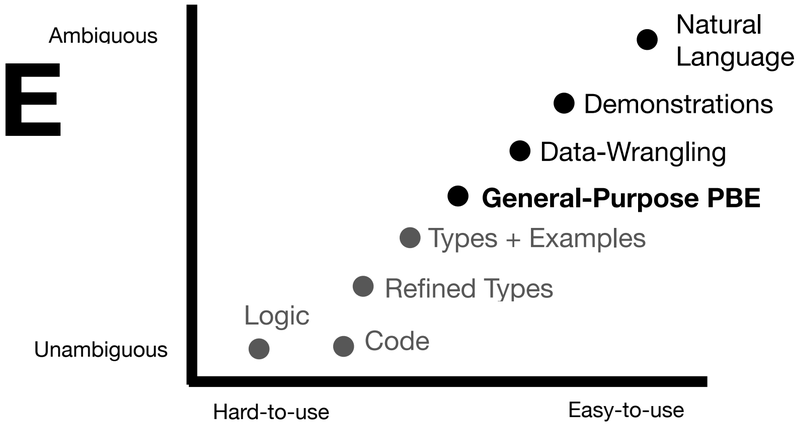
Trade Types
and still cross
the Gulf of Execution?

Yes, but with *many* examples



Input Styles - General Purpose PBE

Trace Completeness



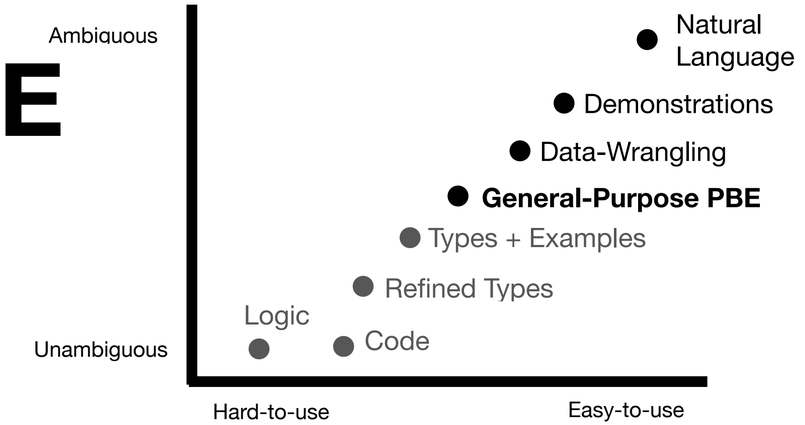

```
stutter “abc” = “aabbcc”

stutter “bc” = “bbcc”

stutter “c” = “cc”

stutter “” = “”
```

The requirement to provide input-output examples for recursive calls internal to the eventual solution

Lubin et al. “Program Sketching with Live Bidirectional Evaluation”. ICFP2020

Input Styles - General Purpose PBE

Trace Completeness

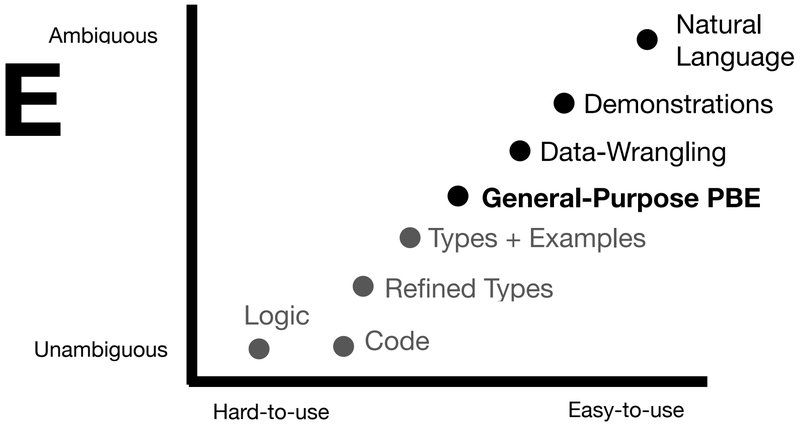



```
stutter “abc” = “aabbcc”

stutter “bc” = “bbcc”

stutter “c” = “cc”

stutter “” = “”
```

Need a conceptual model
to come up with the examples!

Input Styles - General Purpose PBE

Trace Completeness



Work by Lubin et al., removed requirement

```
stutter “abc” = “aabbcc”

stutter “bc” = “bbcc”

stutter “c” = “cc”

stutter “” = “”
```

Input Styles - General Purpose PBE

Trace Completeness

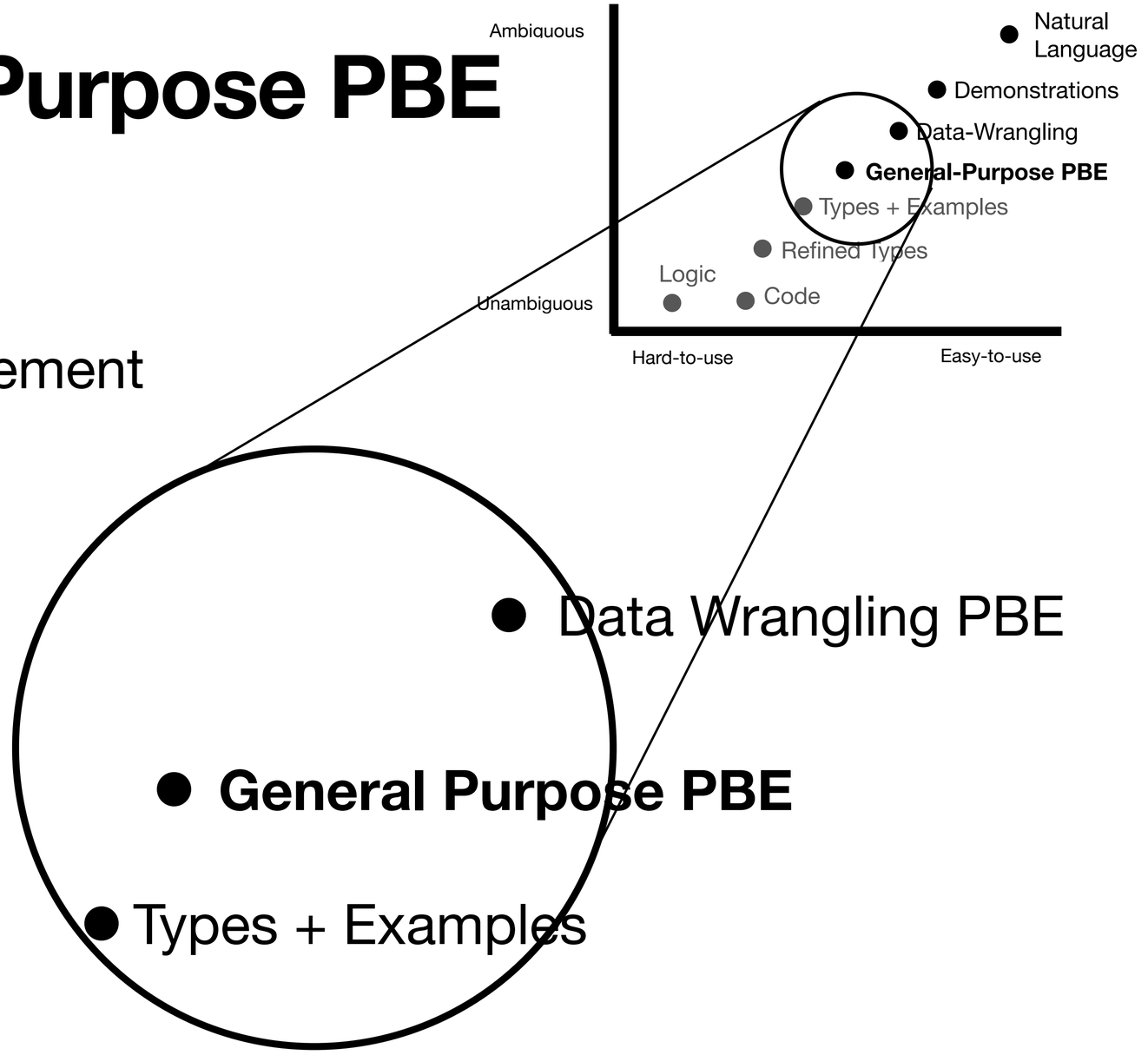
Work by Lubin et al., removed requirement

~~`stutter “abc” = “aabbcc”`~~

`stutter “bc” = “bbcc”`

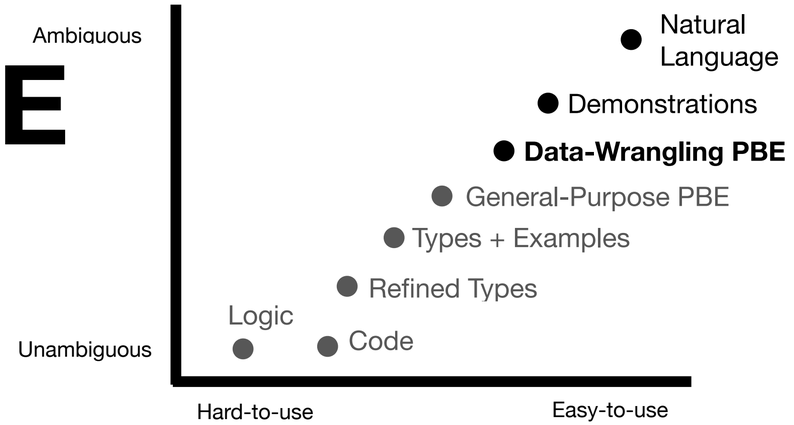
~~`stutter “c” = “cc”`~~

`stutter “” = “”`



Input Style - Data Wrangling PBE

Domain Specific



	A	B	C
1	Data	Currency	Value
2	USD300		
3	RMB9020		
4	SGD134		
5	HKD289		
6	EUR888		
7	MYR483		
8	KRW2302		

FlashFill

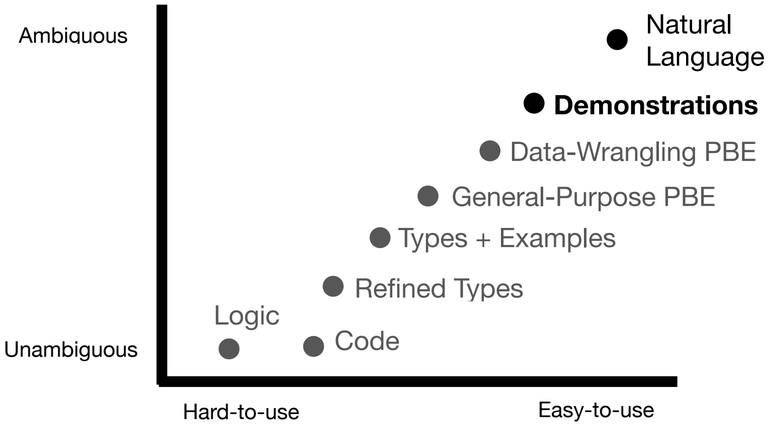
Trading generality for performance

Little friction to using tool

Constrained by setting

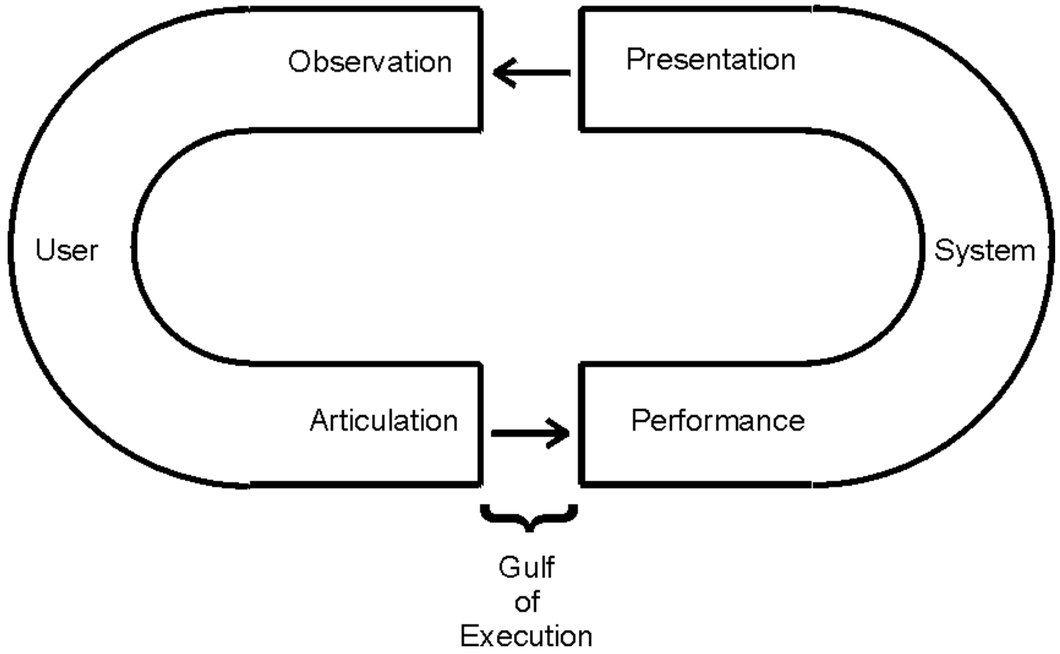
S. Gulwani. “Automating string processing in spreadsheets using input-output examples”. POPL2011

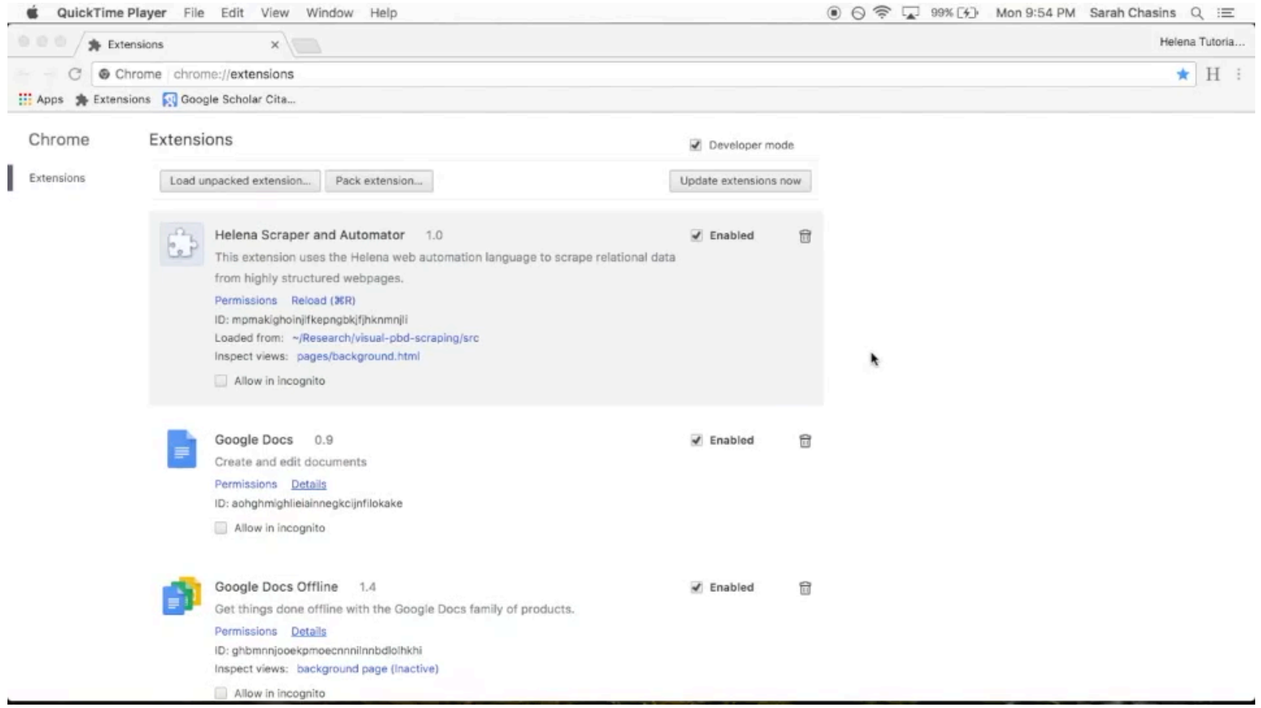
Input Style - Demonstrations



Watch user work

Gulf of Execution is seamless



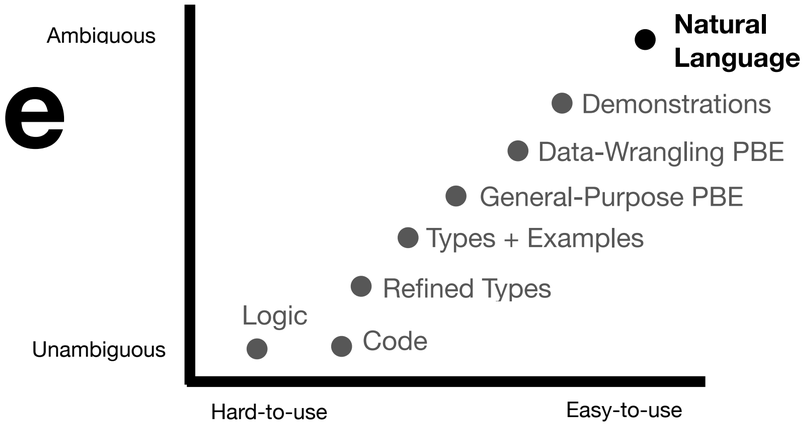


Chasins et al. Roussillon. https://github.com/schasins/helena

Could using synthesizers be as straightforward as asking?

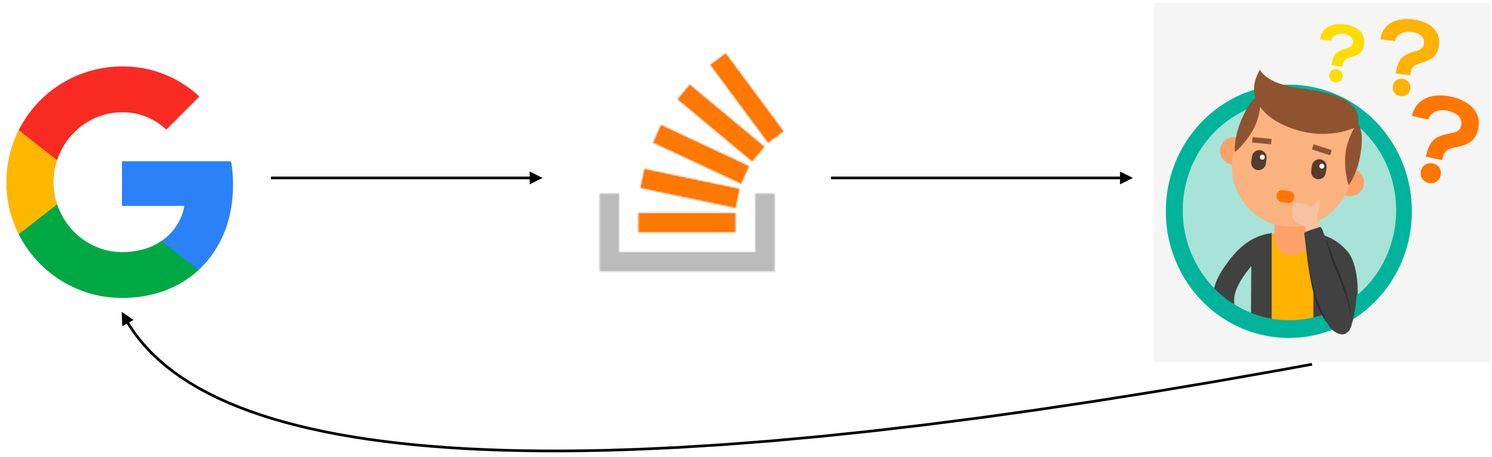
Gulf of Execution:
What can I even ask?

Input Styles - Natural Language

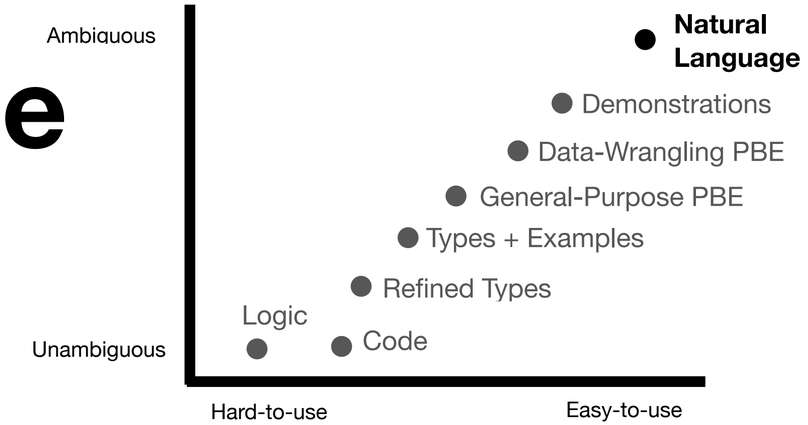


Natural Language does have a place!

Finding a regular expression, normally



Input Styles - Natural Language



Natural Language does have a place!

Finding a regular expression, with Regel:

I need a regular expression to match Decimal(18,3), which means …

Positive Examples

```
12345.1
123
```

Negative Examples

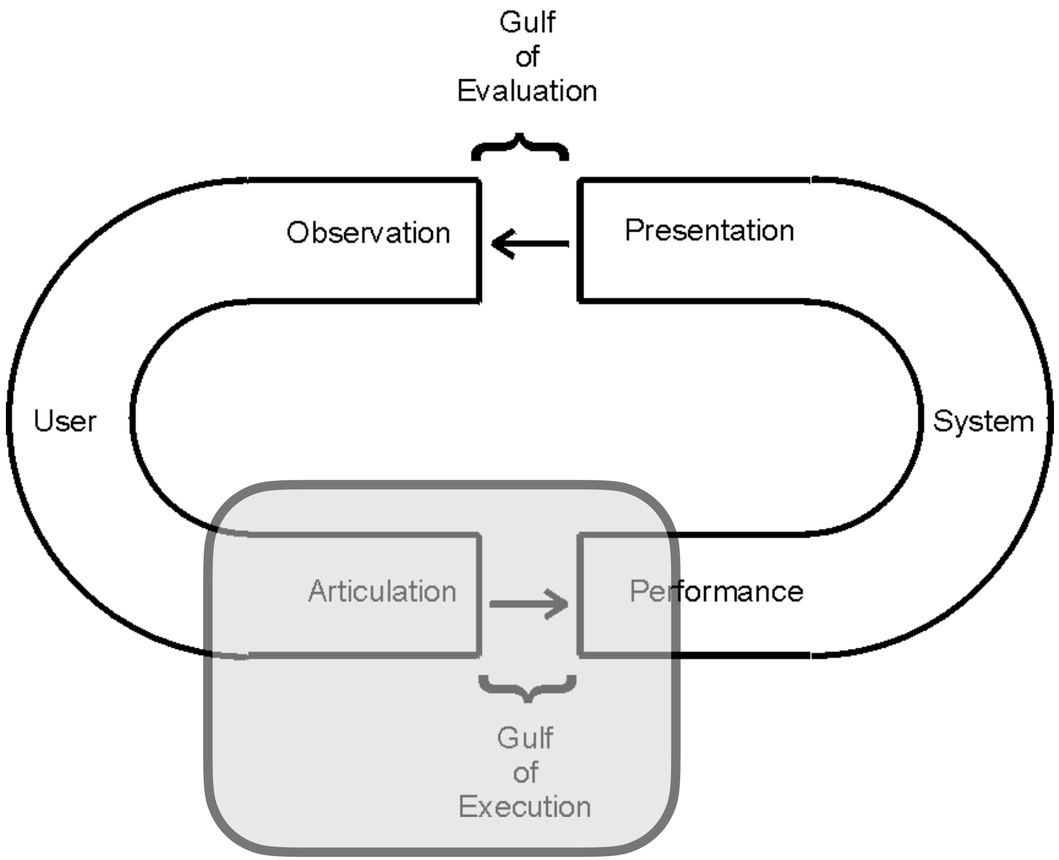
```
1.12345
.1234
```

[1]
Q. Chen, X. Wang, X. Ye, G. Durrett, and I. Dillig, "Multi-modal synthesis of regular expressions," in PLDI 2020, doi: 10.1145/3385412.3385988.

Input - Open Questions

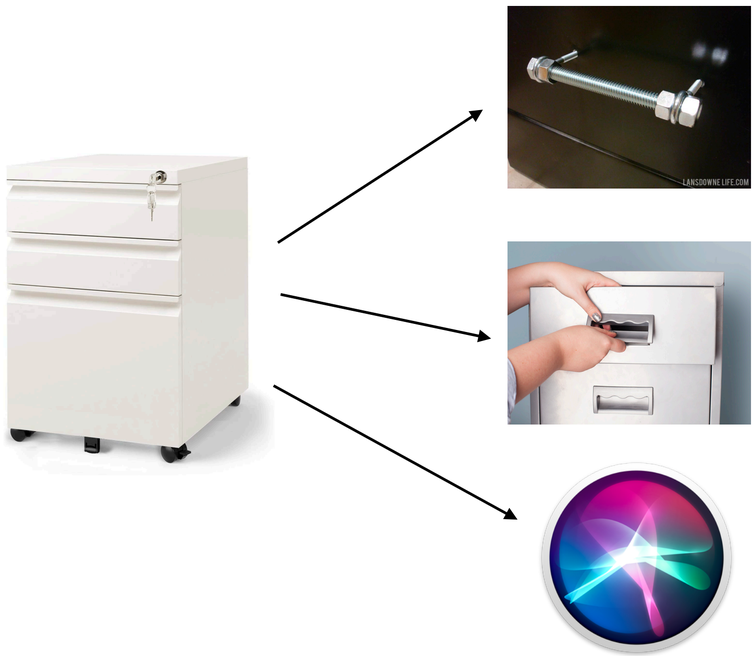
What are user preferences, across tasks?

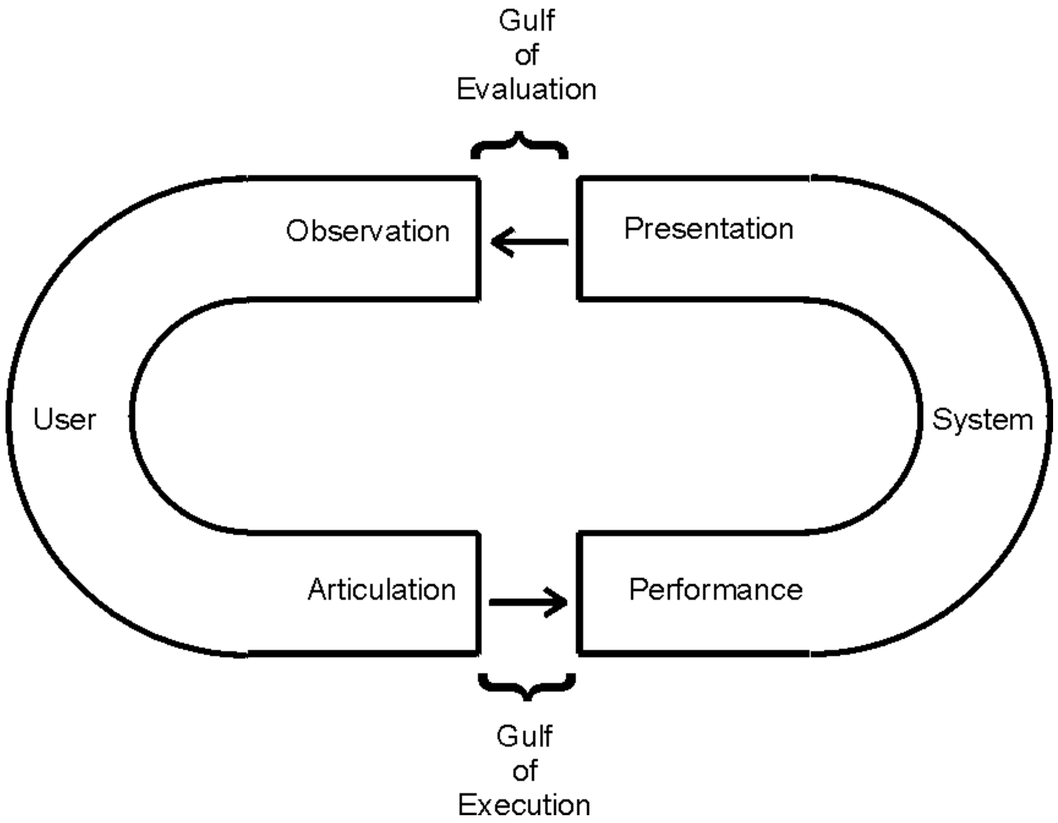
When has a user provided enough input?



Input - Open Questions

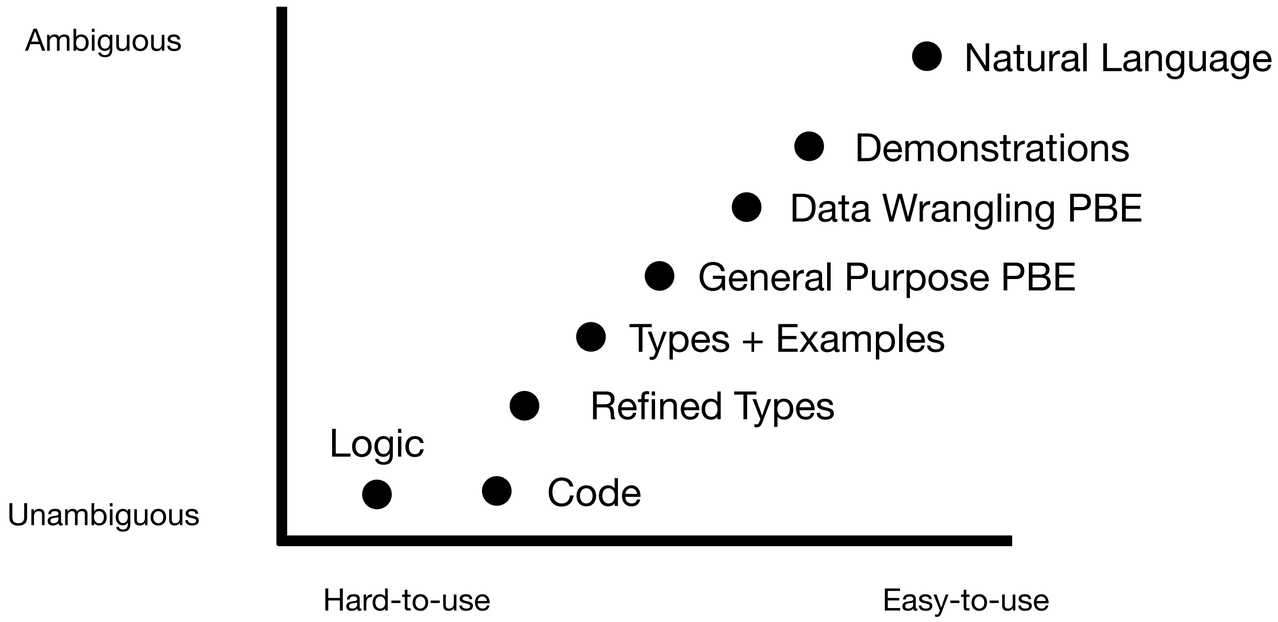
What are user preferences, across tasks?

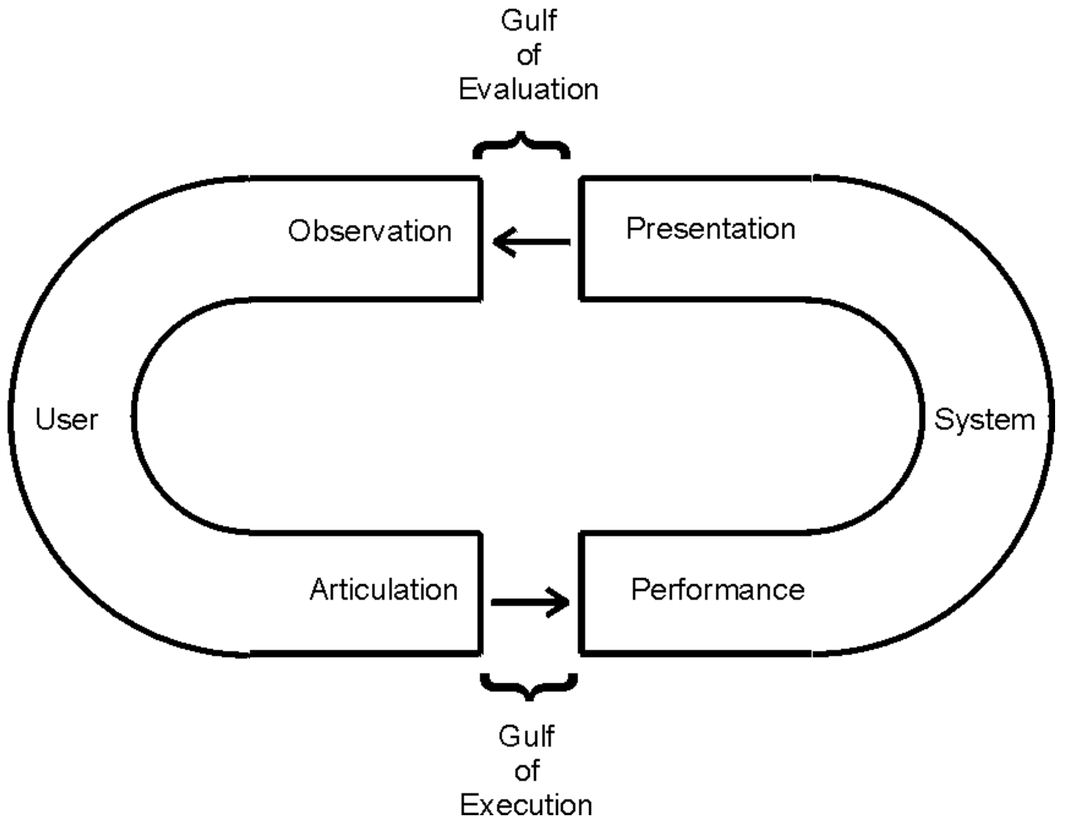




Input - Open Questions

What are user preferences, across tasks?





Input - Open Questions

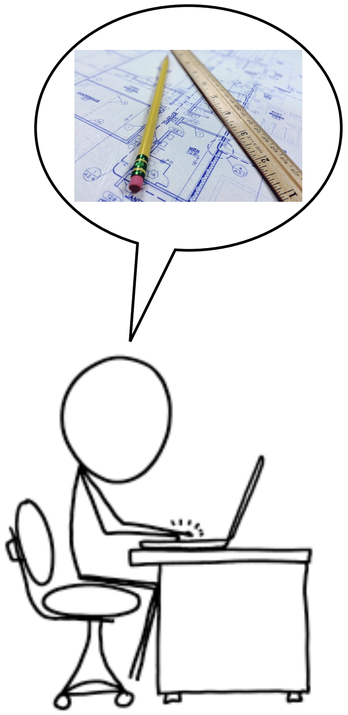
When has a user provided enough input?

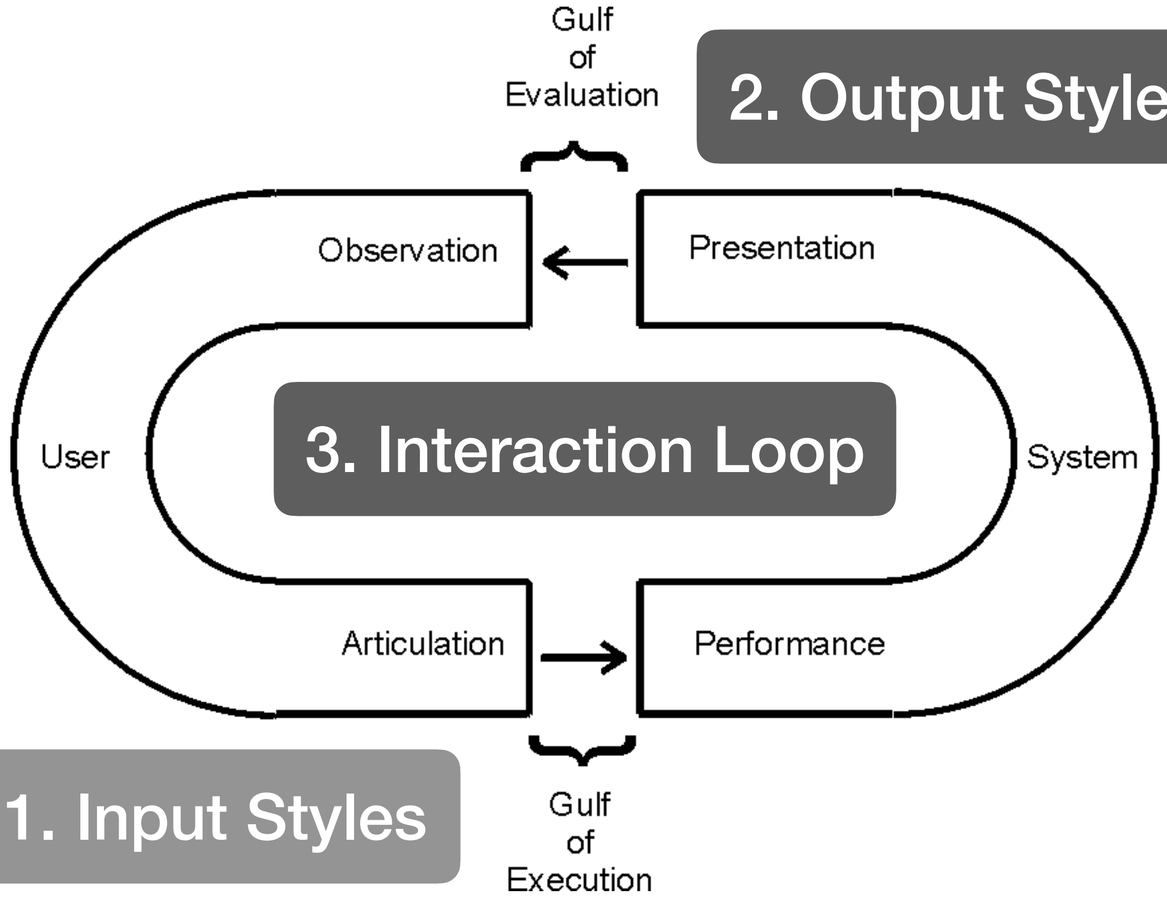
```
stutter :: xs: List a -> {List a | len _v == (len xs) * 2}
```

```
stutter “abc” = “aabbcc”
stutter “bc” = “bbcc”
stutter “c” = “cc”
stutter “” = “”
```

I need a regular expression to match Decimal(18,3), which means …

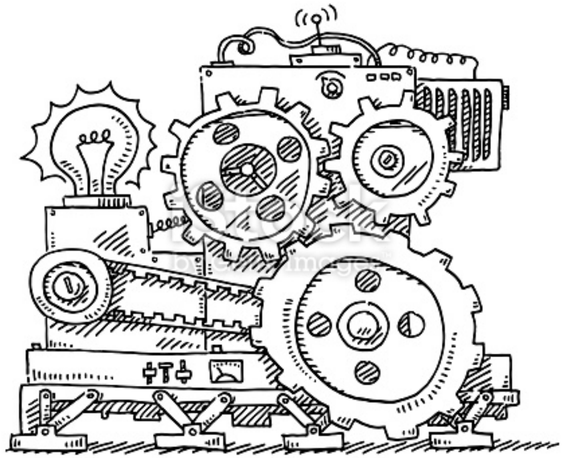
Did I just need to pull harder?





2. Output Styles

3. Interaction Loop



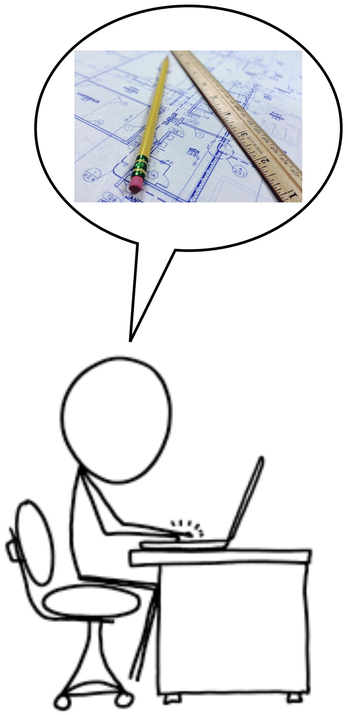
1. Input Styles

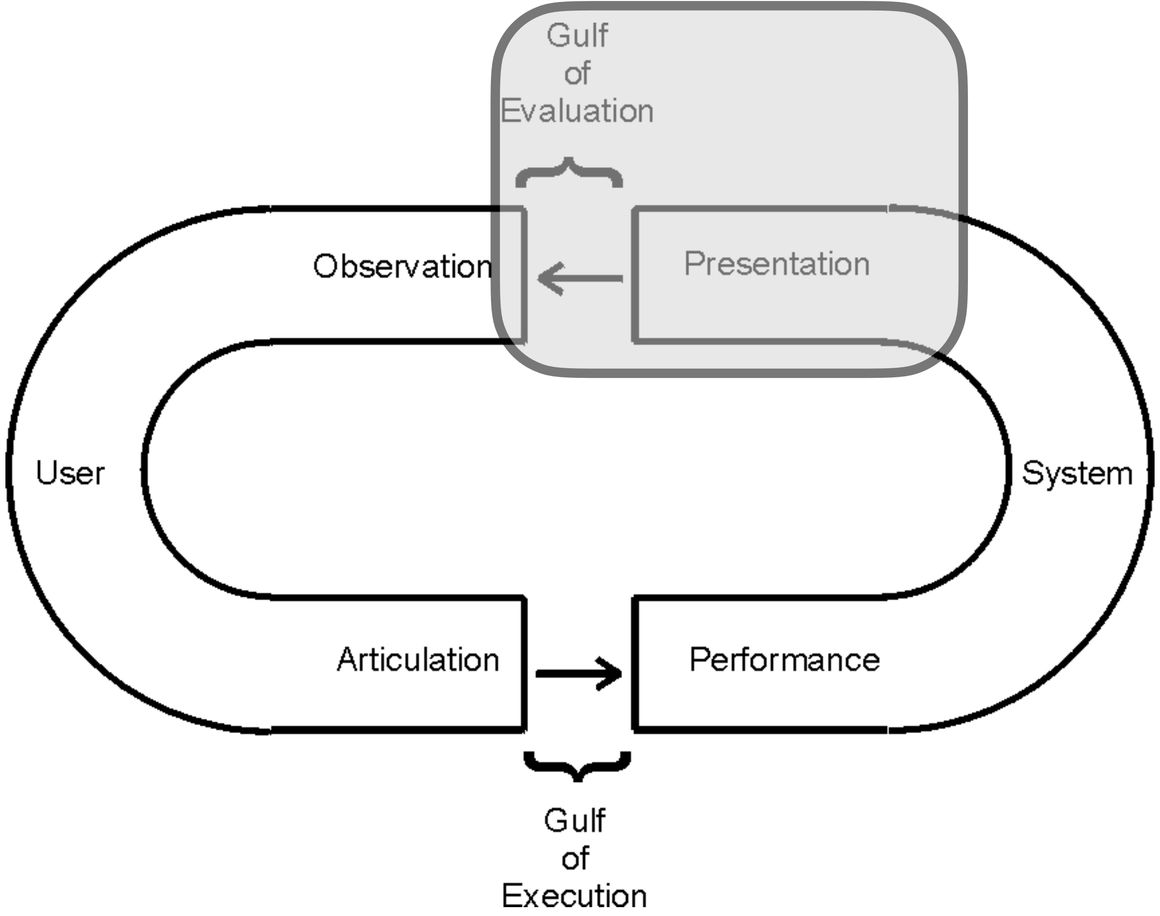
4. My Work

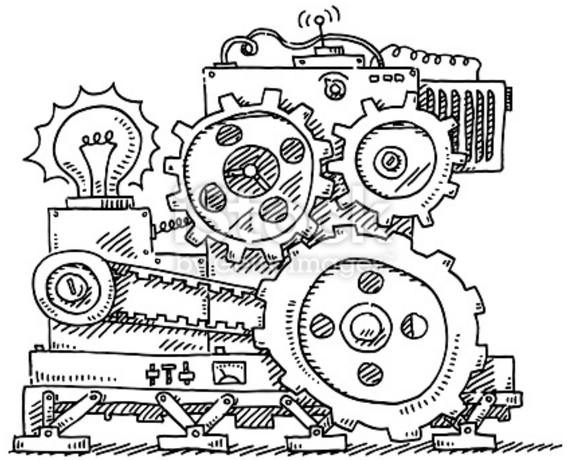
Hoogle+

5. Open Questions

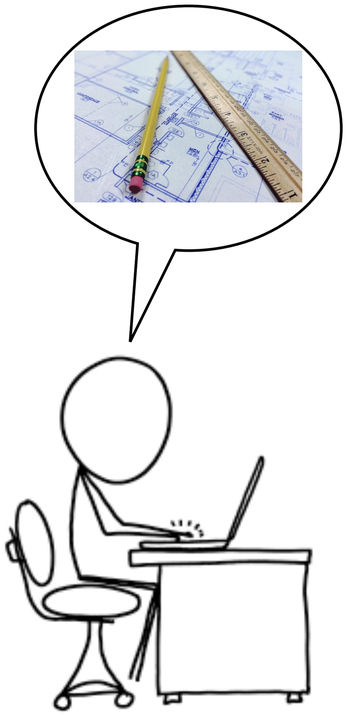
http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

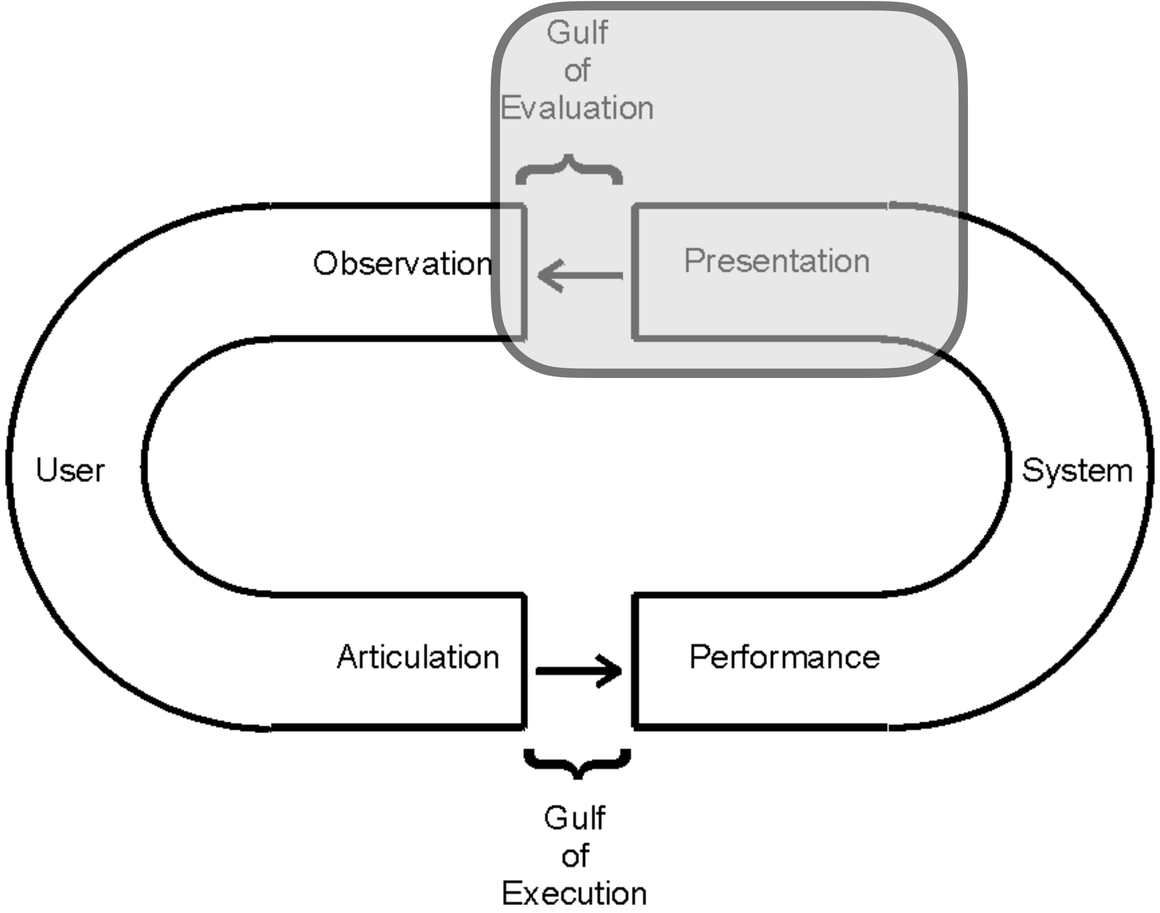


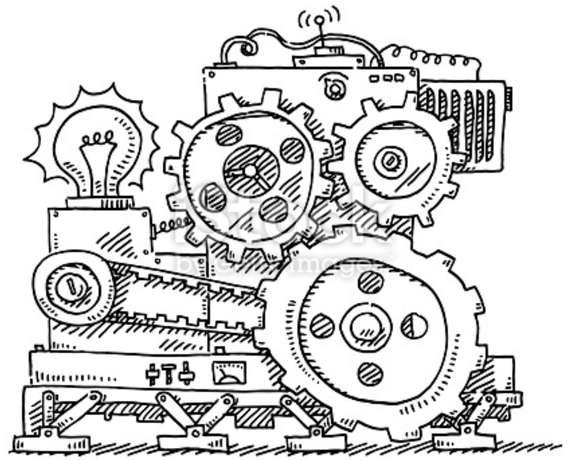




Gulf of Evaluation: Gap between the **language of the system** and the **language of your goal**







Natural Language	Results	Code	Ranking	Comprehension

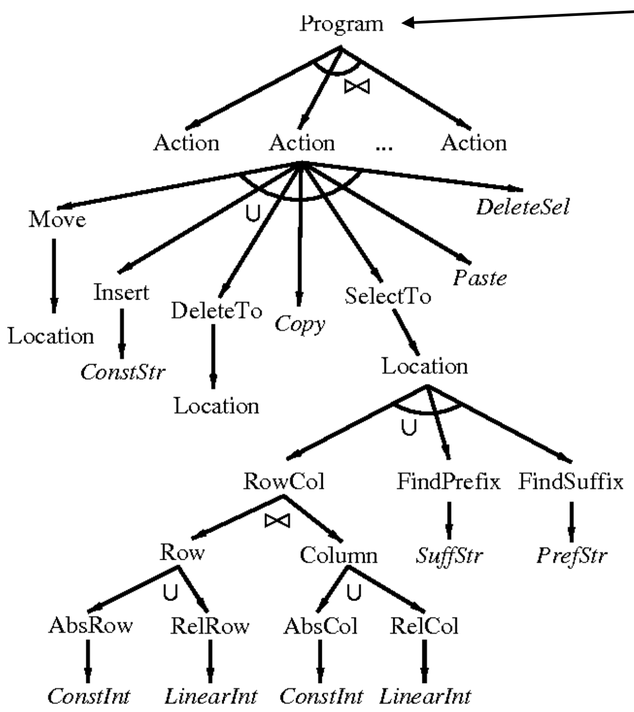
Natural Language

Internal Search Representation  Natural Language

- Typing Rules
- Enumeration State
- Graph Representation
- Version Space Algebra

Natural Language

Version Space Algebra



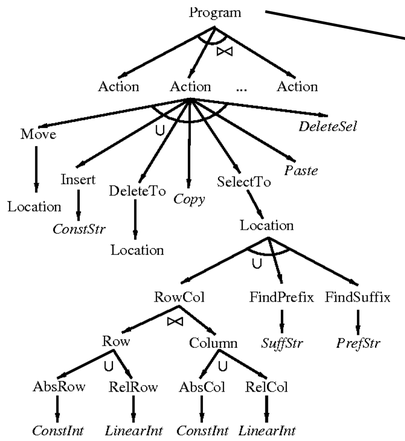
One Program, implemented many ways

Paths represent specific programs

Use rules to generate sentences from path

Natural Language

Paraphrasing each node in the VSA!



To extract AuthorList from Record:
From the substring starting at the first occurrence of end of WhiteSpace,
extract the string ending at
the first occurrence of
end of Camel Case
in the second line

FlashProg

Mayer et al. “User Interaction Models for Disambiguation in Programming by Example”. UIST 2015

Execution Result

- Just do it
- Immediately usable
- Immediate feedback

- One-time-use
- Not scalable

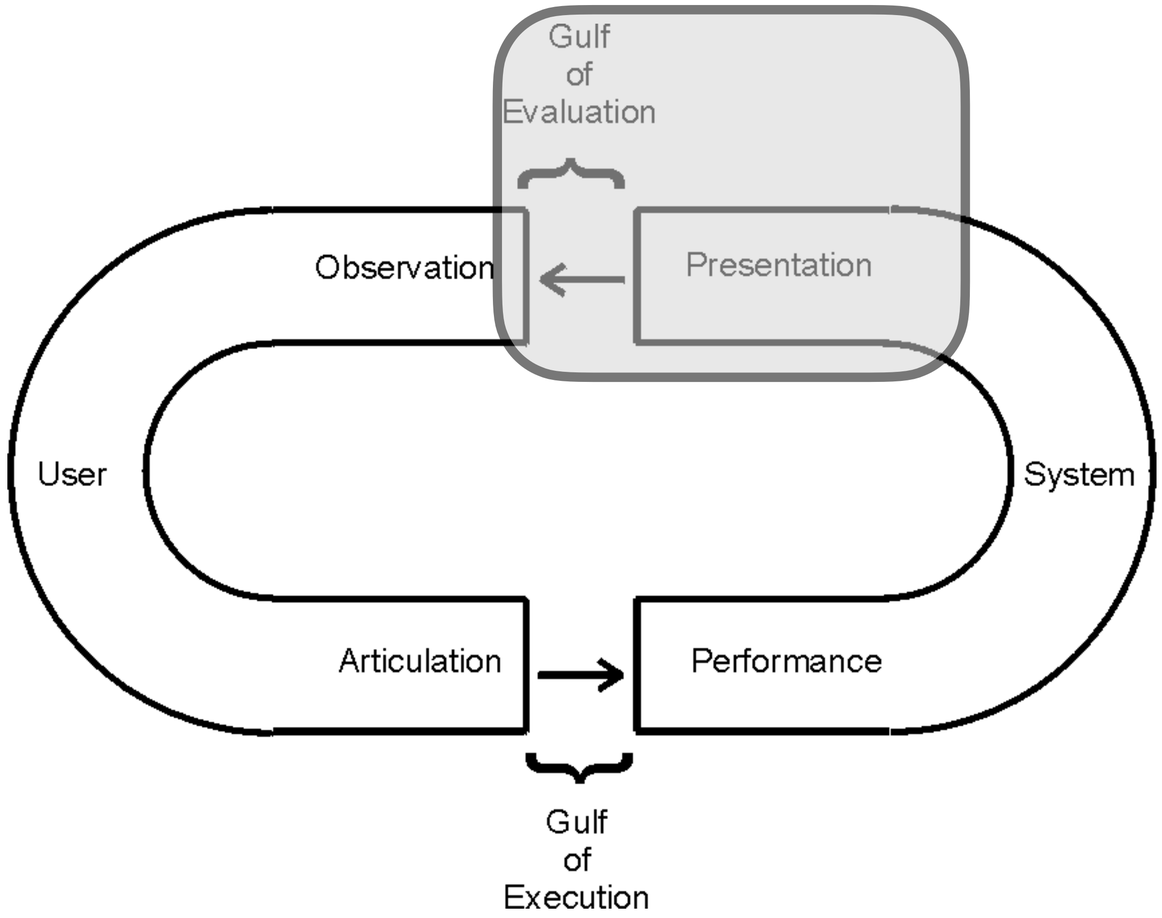
	A	B	C
1	Data	Currency	Value
2	USD300		
3	RMB9020		
4	SGD134		
5	HKD289		
6	EUR888		
7	MYR483		
8	KRW2302		

Flashfill

Code

Familiar to Programmers

Hard to translate to language of goal

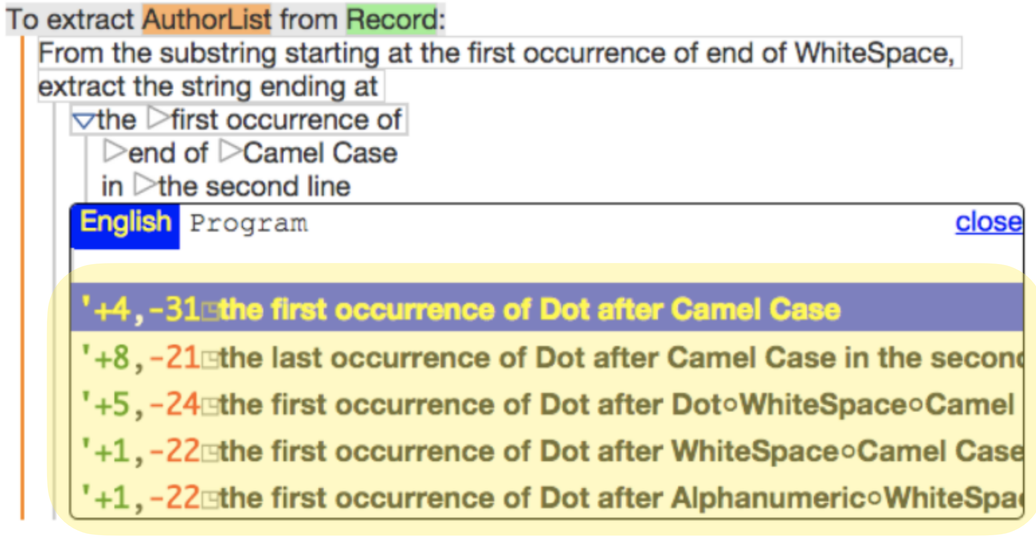


Ranking

Synthesizers can produce many possible programs

Which to show a user first?

- Size: Synquid
- Information Loss
- Complexity: FlashFill-style
- Probabilities

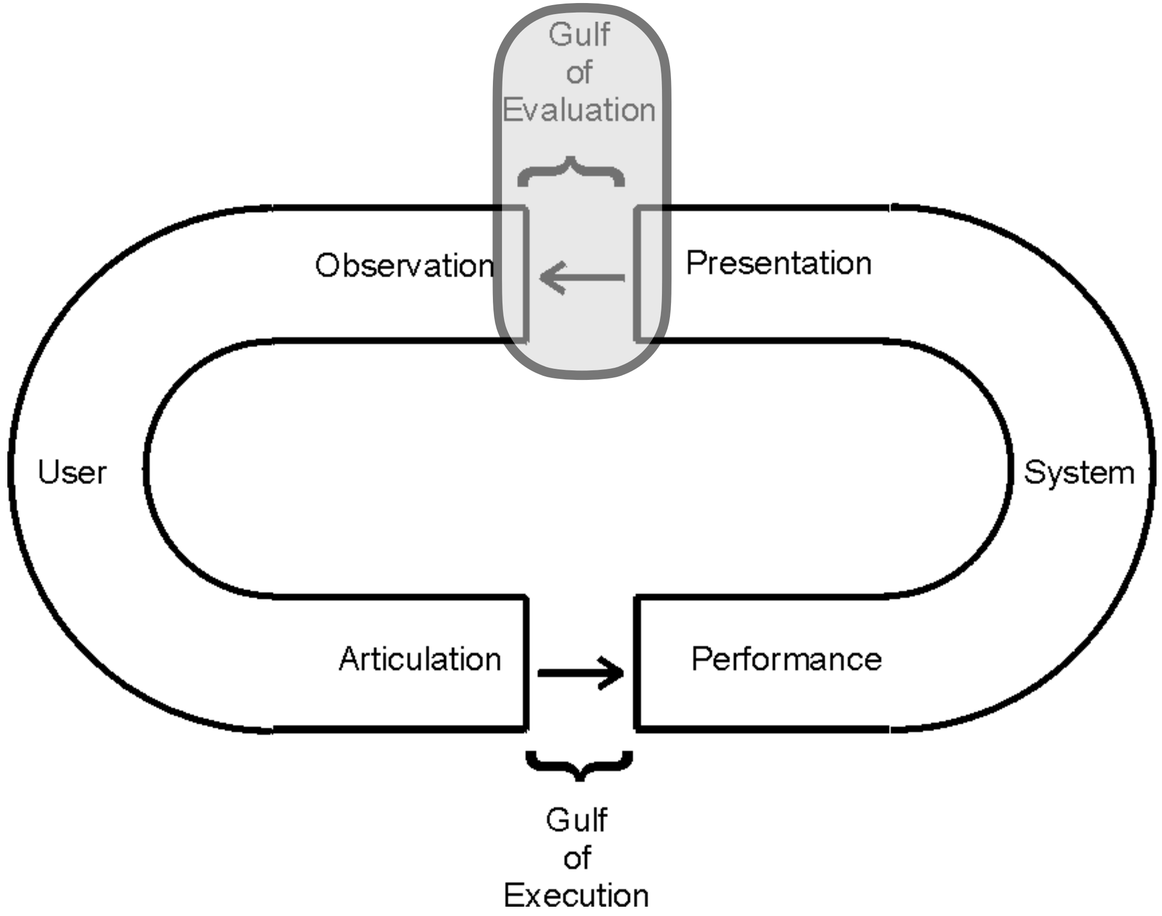


Mayer et al. “User Interaction Models for Disambiguation in Programming by Example”. UIST 2015

Comprehension

Will users recognize the right answer when it’s given?

```
DEBUG: Property: Company
Program: ESSL((EndsWith(Dynamic Token(</td><td>)(</td><td>), ALL CAPS(\p{Lu}(\p{Lu})+),
Dynamic Token(</td></tr>)(</td></tr>))): 0,
1, ...: Dynamic Token(<tr><td>)(<tr><td>)...Alphabet([\p{Lu}\p{Ll}\-.]+), Dynamic
Token(</td><td>)(</td><td>), ALL CAPS(\p{Lu}(\p{Lu
})+), 1 + Camel Case(\p{Lu}(\p{Ll})+)...Dynamic Token(</td><td>)(</td><td>), ALL
CAPS(\p{Lu}(\p{Lu})+), Dynamic Token(</td></tr>)(</
td></tr>), 1)
```



Comprehension

Intermediate Values

Seeing Effect of code (I/O Examples)

Sensible Variable Naming

Readable Code

Task: Get most common bigram

Read code down

code	Debug information (example 1)
input	"abdfibfcfdebdfdebdihgfkjfdebd"
zip(input.tail)	List((a,b),(b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),
drop(1)	List((b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),(d,e),
map(p => p._1.toString + p._2)	List("bd","df","fi","ib","bf","fc","cf","fd","de","eb","bd",
min	"bd"

H. Peleg et al. “Programming Not Only By Example”. ICSE 2018

Comprehension

Intermediate Values

Seeing Effect of code (I/O Examples)

Sensible Variable Naming

Readable Code

Task: Get most common bigram

Read code down

code	Debug information (example 1)
input	"abdfibfcfdebdfdebdihgfkjfdebd"
zip(input.tail)	List((a,b),(b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),
drop(1)	List((b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),(d,e),
map(p => p._1.toString + p._2)	List("bd","df","fi","ib","bf","fc","cf","fd","de","eb","bd",
min	"bd"

H. Peleg et al. “Programming Not Only By Example”. ICSE 2018

Comprehension

Intermediate Values

Seeing Effect of code (I/O Examples)

Sensible Variable Naming

Readable Code

Task: Get most common bigram

Read code down

code	Debug information (example 1)
input	"abdfibfcfdebdfdebdihgfkjfdebd"
zip(input.tail)	List((a,b),(b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),
drop(1)	List((b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),(d,e),
map(p => p._1.toString + p._2)	List("bd","df","fi","ib","bf","fc","cf","fd","de","eb","bd",
min	"bd"

H. Peleg et al. “Programming Not Only By Example”. ICSE 2018

Comprehension

Intermediate Values

Seeing Effect of code (I/O Examples)

Sensible Variable Naming

Readable Code

Task: Get most common bigram

Read code down

code	Debug information (example 1)
input	"abdfibfcfdebdfdebdihgfkjfdebd"
zip(input.tail)	List((a,b),(b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),
drop(1)	List((b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),(d,e),
map(p => p._1.toString + p._2)	List("bd","df","fi","ib","bf","fc","cf","fd","de","eb","bd",
min	"bd"

H. Peleg et al. “Programming Not Only By Example”. ICSE 2018

Comprehension

Task: Get most common bigram

Read code down

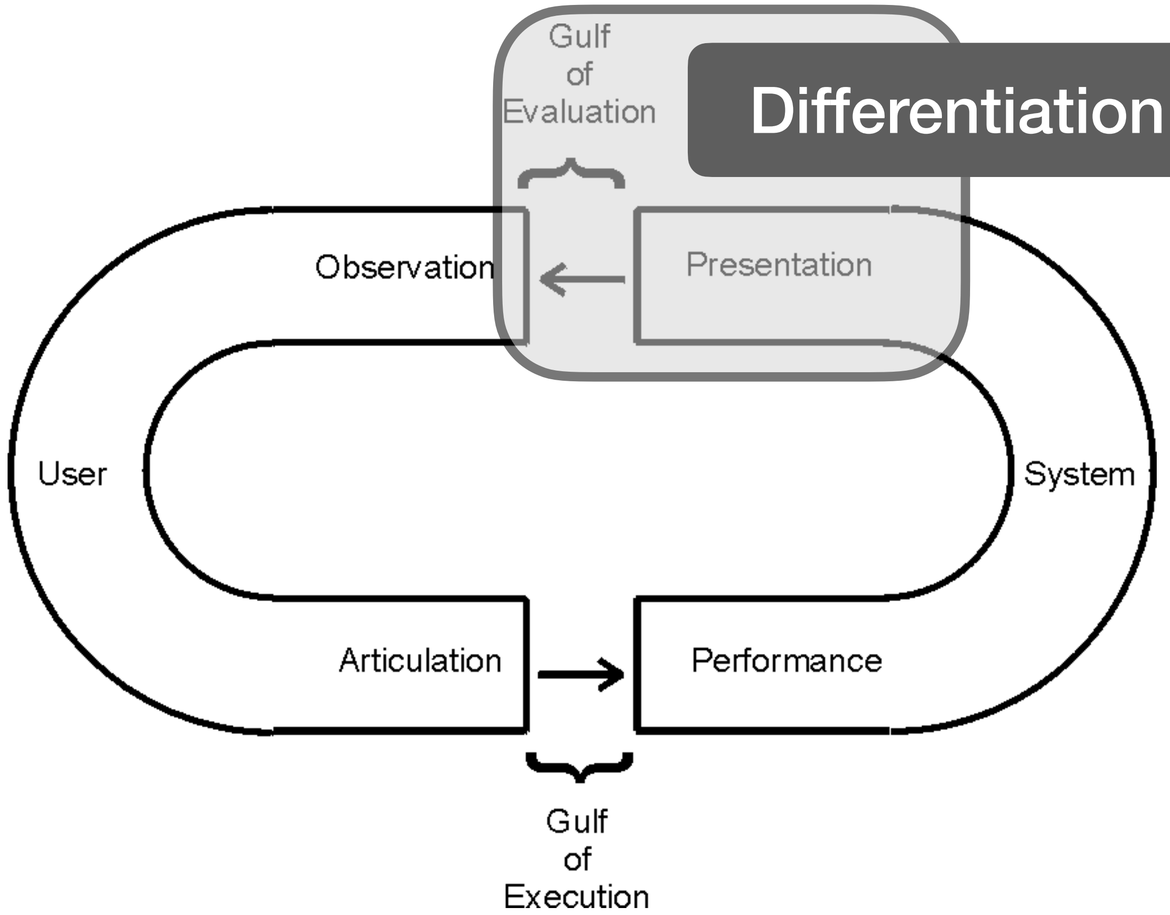
code	Debug information (example 1)
input	"abdfibfcfdebdfdebdihgfkjfdebd"
zip(input.tail)	List((a,b),(b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),
drop(1)	List((b,d),(d,f),(f,i),(i,b),(b,f),(f,c),(c,f),(f,d),(d,e),
map(p => p._1.toString + p._2)	List("bd","df","fi","ib","bf","fc","cf","fd","de","eb","bd",
min	"bd"

Vs

```
input.zip(input.tail).drop(1).map(p => p._1.toString + p._2).min
```

H. Peleg et al. “Programming Not Only By Example”. ICSE 2018

Open Questions

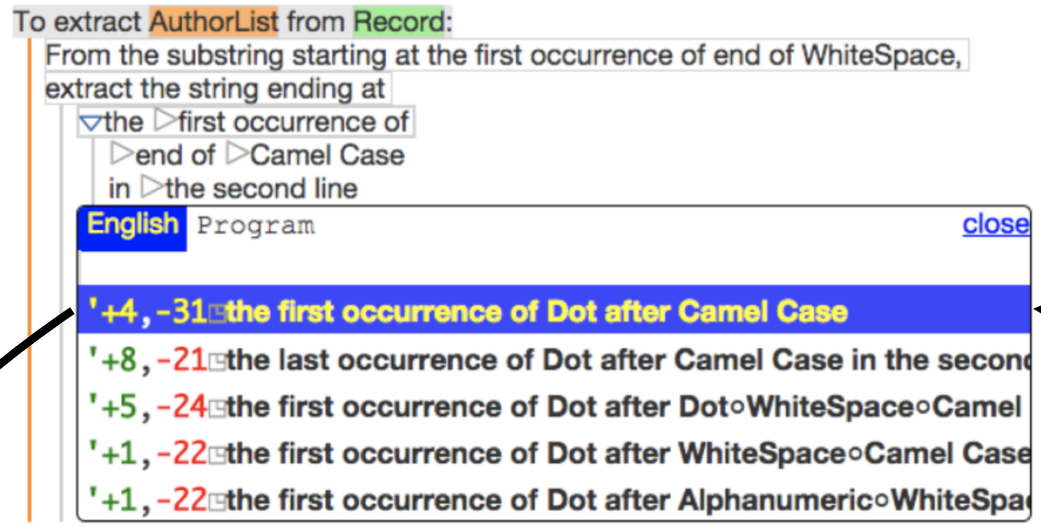


Differentiation

Impatience

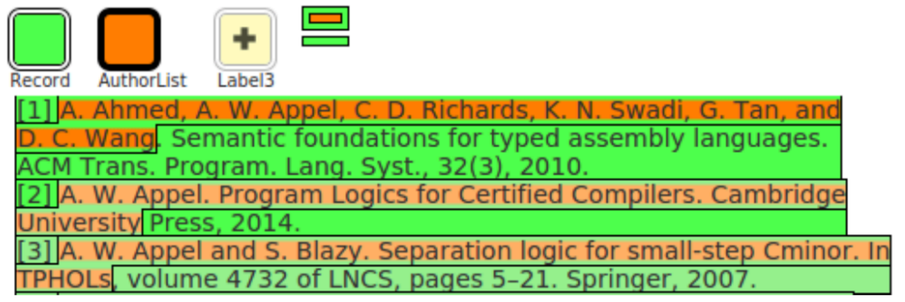
Open Question

Differentiation



What's the difference between these two?

Hover to see effect

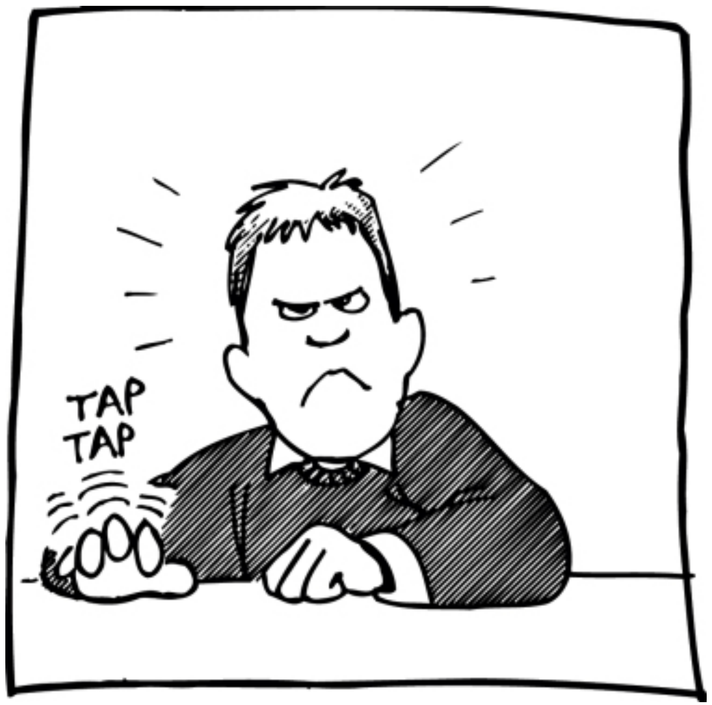


Mayer et al. "User Interaction Models for Disambiguation in Programming by Example". UIST 2015

Open Question

Impatience

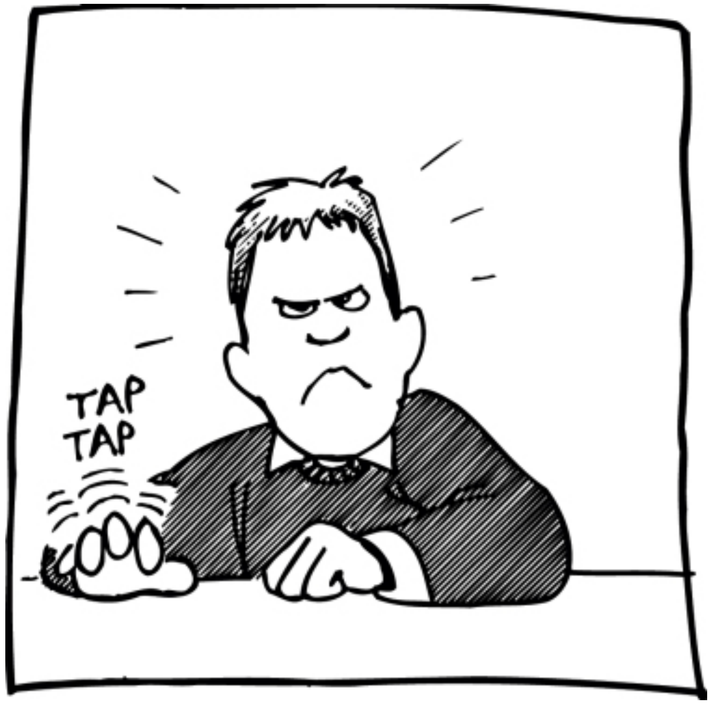
How long would you wait?



Open Question

Impatience

How long would you wait?

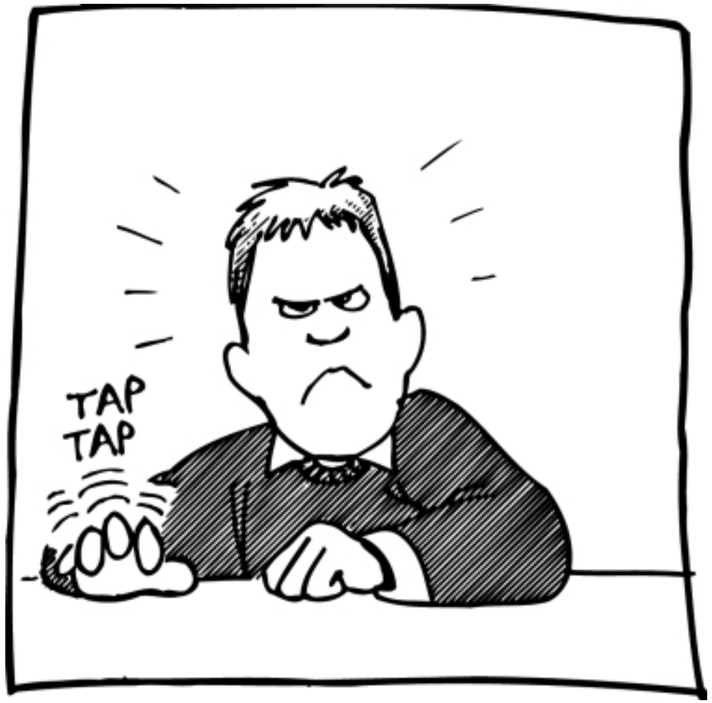


< 1 second?
10 seconds?
1 minute?

Open Question

Impatience

How long would you wait?



How long *can* you wait?

At least 30 seconds

Oulasvirta and Saarilouoma 2006

Open Question

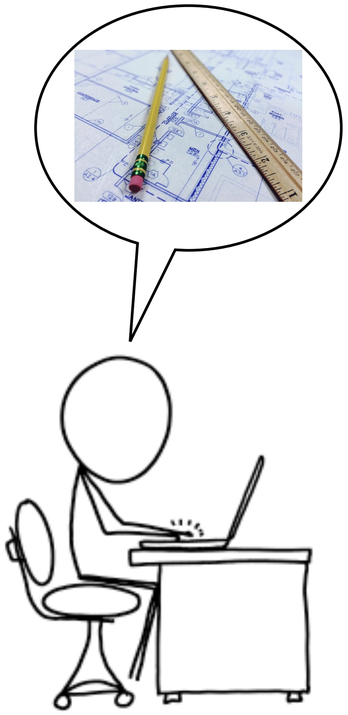
Impatience

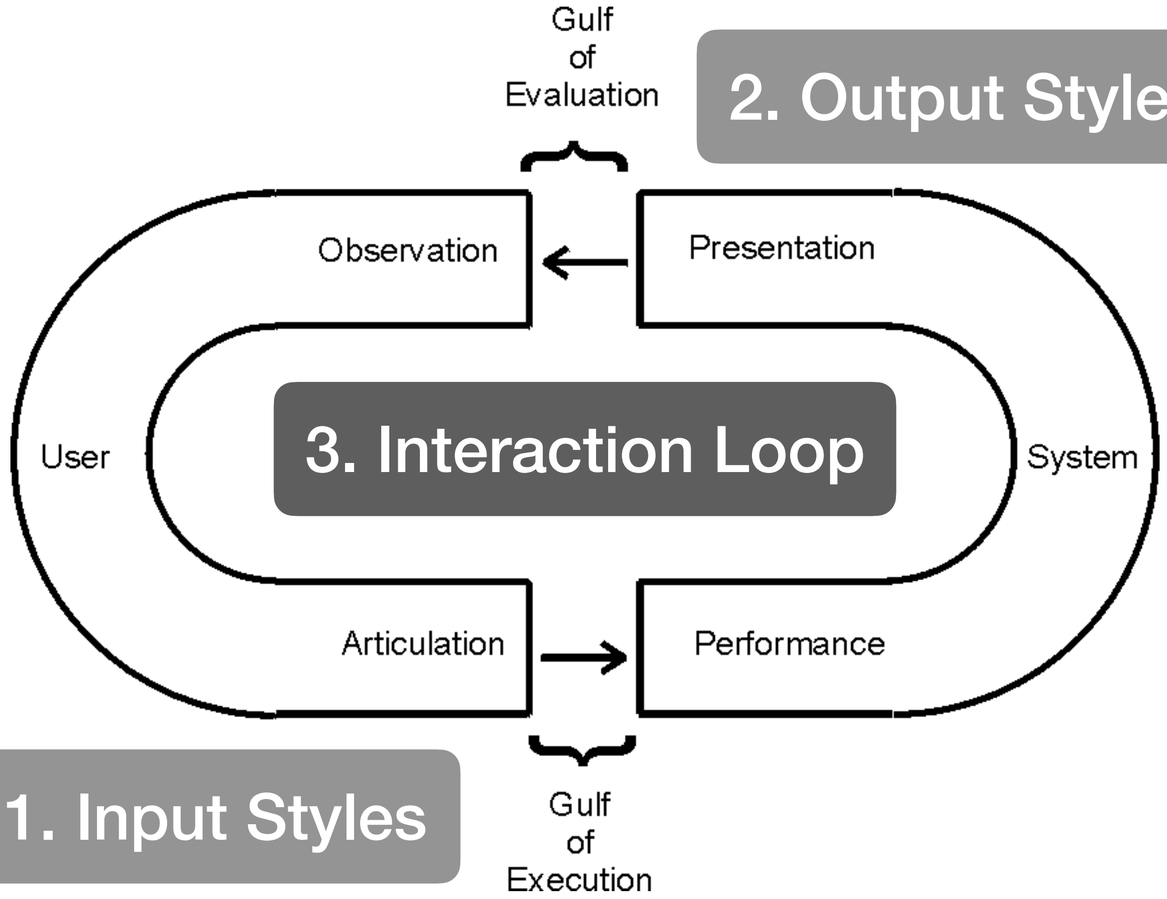
Is there meaningful feedback we can give?

Can users choose accuracy over time?

Can users do something while waiting?

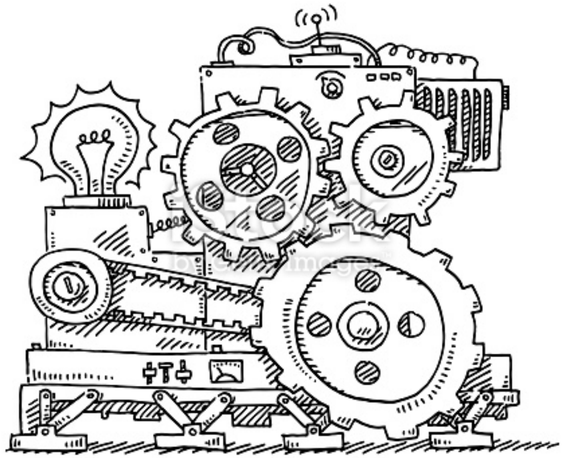
Oulasvirta and Saarilouoma 2006





2. Output Styles

3. Interaction Loop



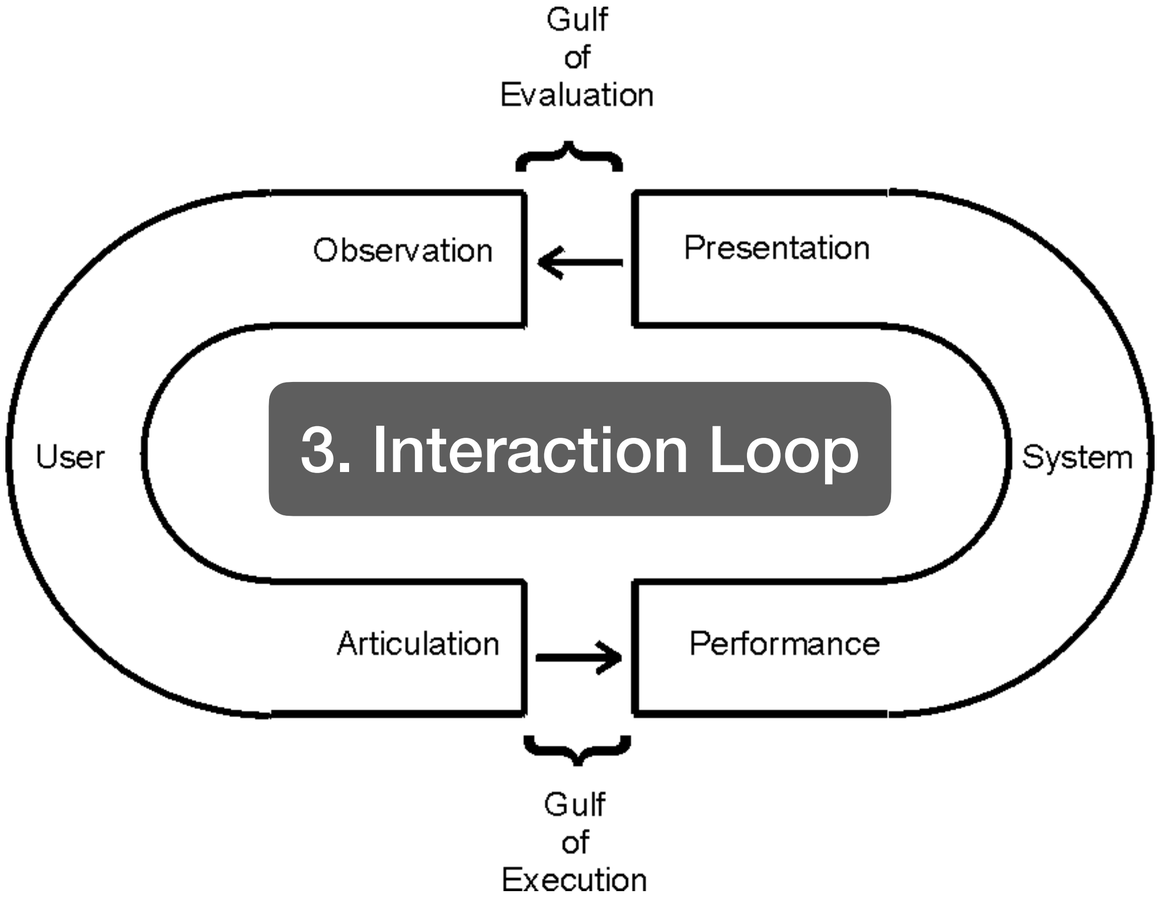
1. Input Styles

4. My Work

Hoogle+

5. Open Questions

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/



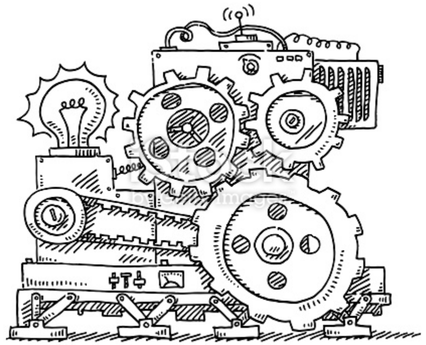
1st loop

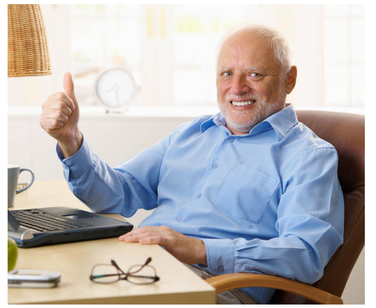
2nd loop

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

Interaction Loop

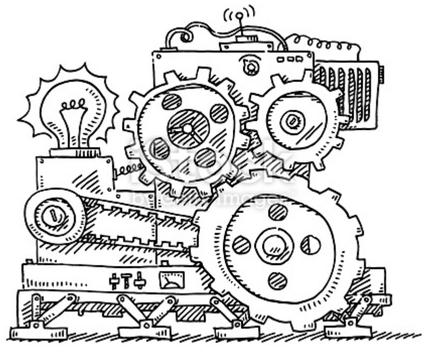
Interactive Program Synthesis

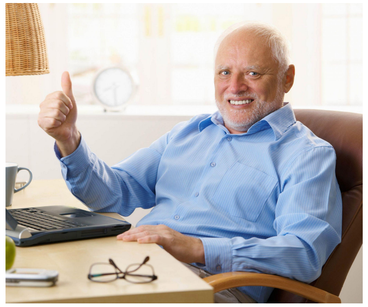




New Information

Soliciting Specifications





Is it useful?

Who refines the search?

User-Driven

Synthesizer-Driven

Granular Interaction

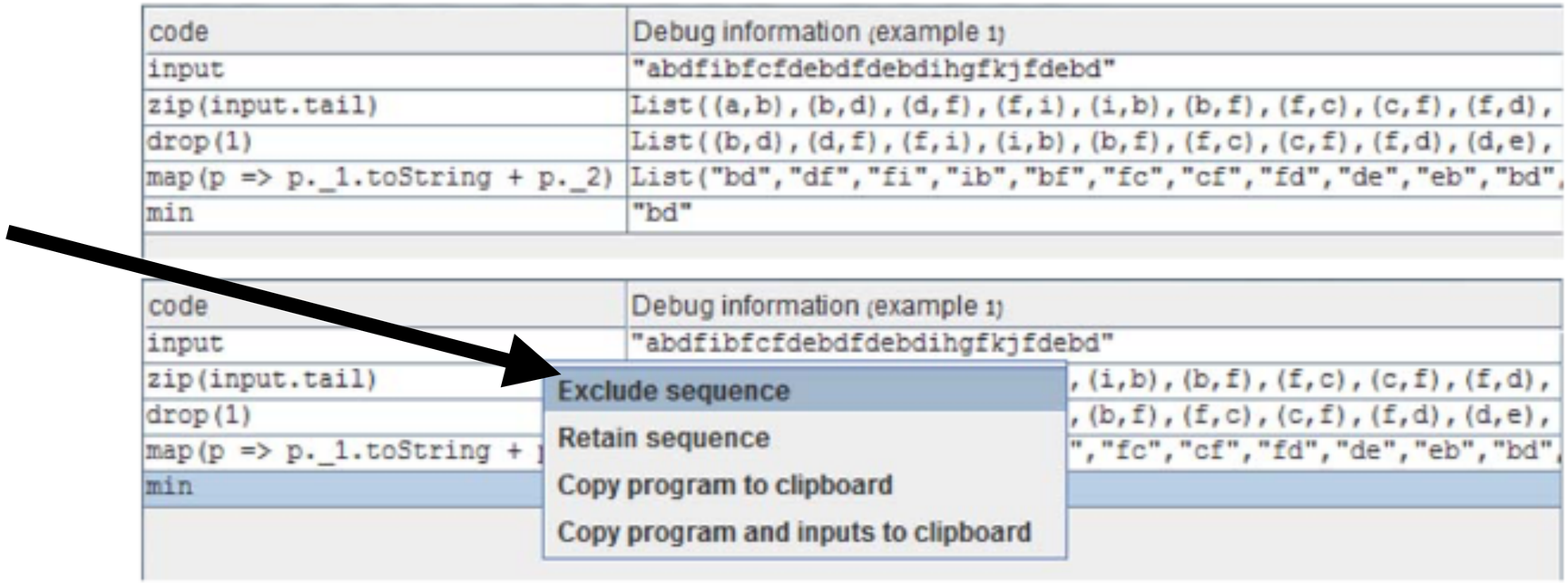
Freeform Examples

Disambiguating Example

Granular Interaction

User-Driven

Task: find most common bigram
must rule out these programs



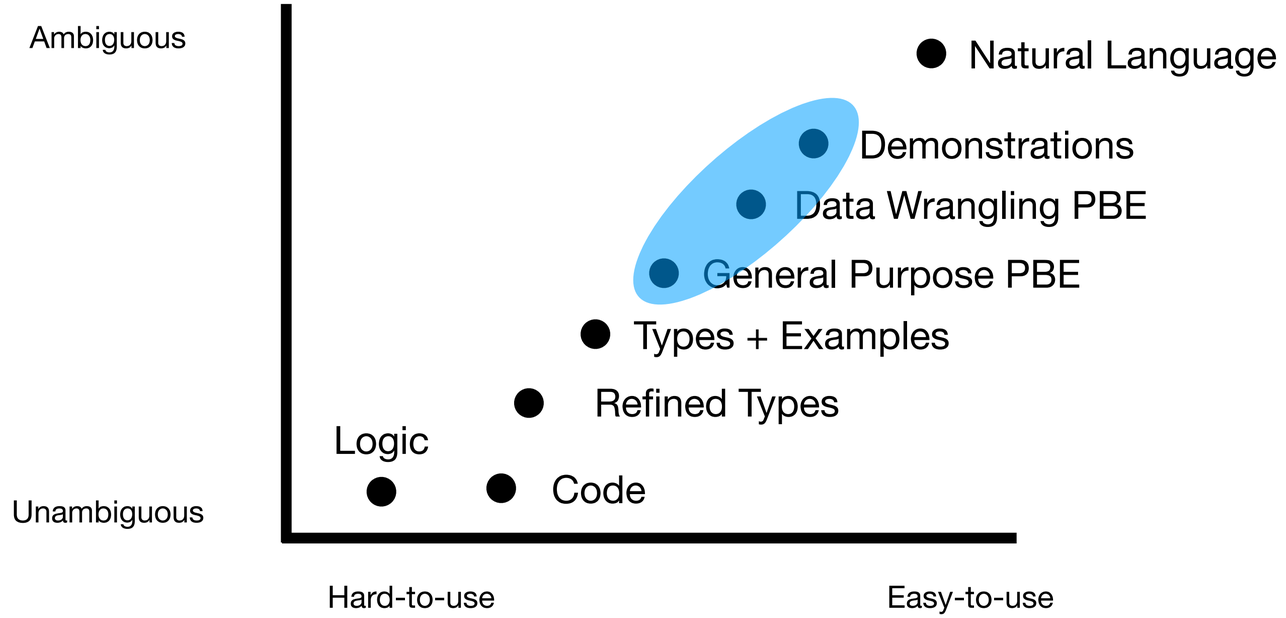
H. Peleg et al. “Programming Not Only By Example”. ICSE 2018

Freeform Examples

User-Driven

Provide another demonstration / example

Full Name	Last Name
Angie McKue	McKue
Lucina Lentsch	Lentsch
Katlin Babidge	Babidge
Karla Rolse	Rolse
Carl Deverille	Deverille



Freeform Examples

User-Driven

Provide another demonstration / example

Full Name	Last Name
Angie McKue	McKue
Lucina Lentsch	Lentsch
Katlin Babidge	Babidge
Karla Rolse	Rolse
Carl Deverille	Deverille

No guarantee search will improve

Freeform Examples

User-Driven

No guarantee search will improve

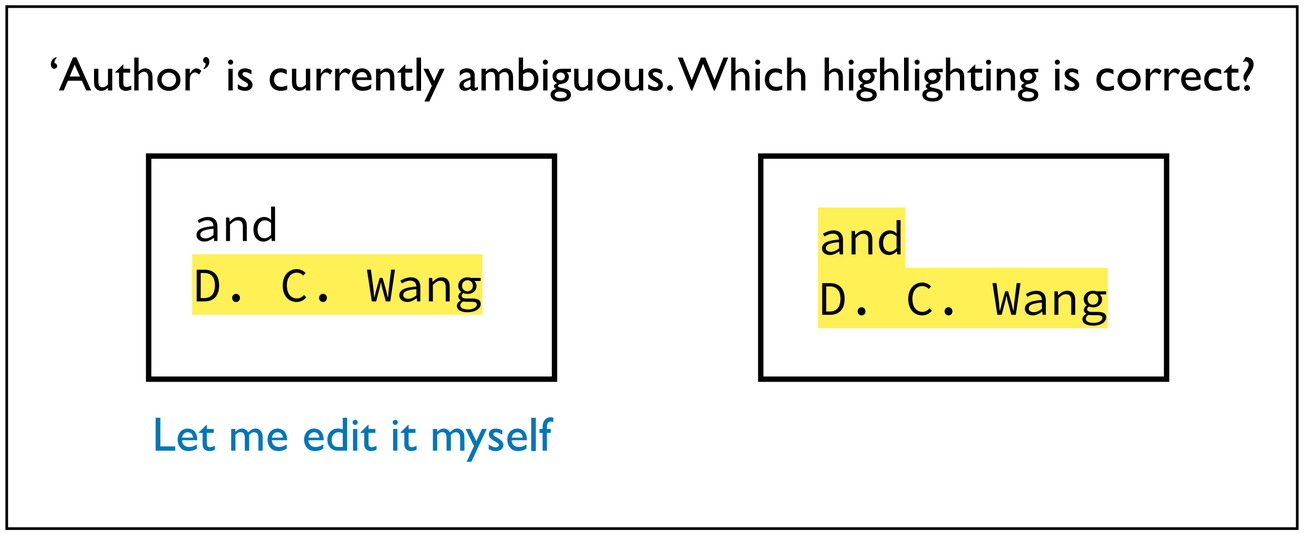
How can we force progress?

Distinguishing Input

Synthesizer-Driven

An input whose output distinguish between at least 2 programs

Task: Highlight author names in semi-structured data



FlashProg (2015)

Mayer et al. "User Interaction Models for Disambiguation in Programming by Example". UIST 2015

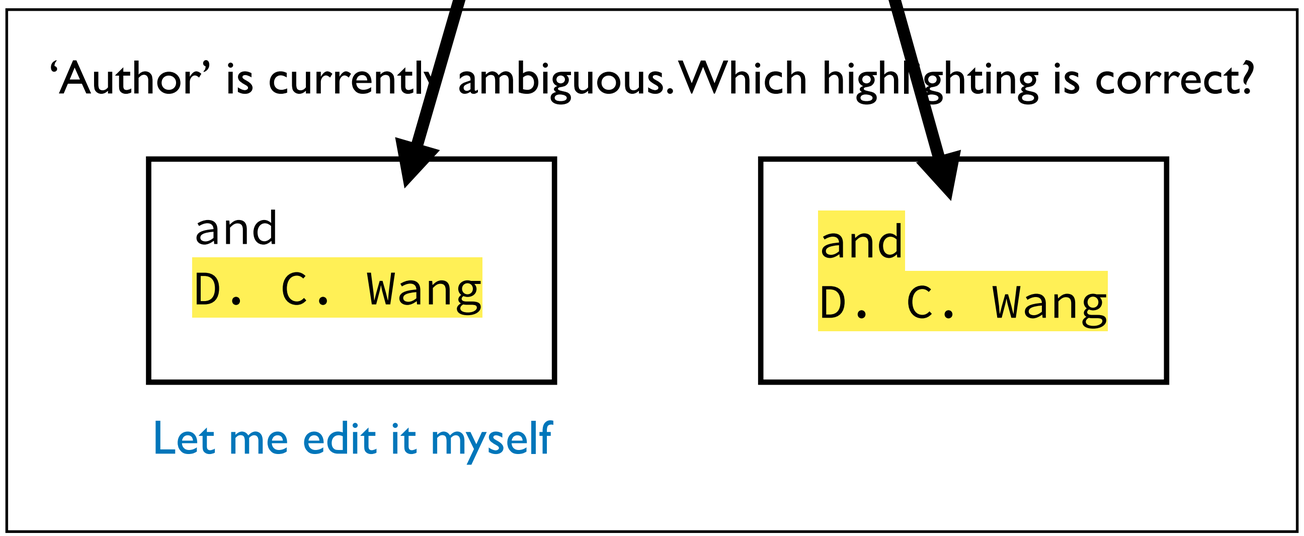
Distinguishing Input

Synthesizer-Driven

An input whose output distinguish between at least 2 programs

Where did these come from?

Task: Highlight author names in semi-structured data

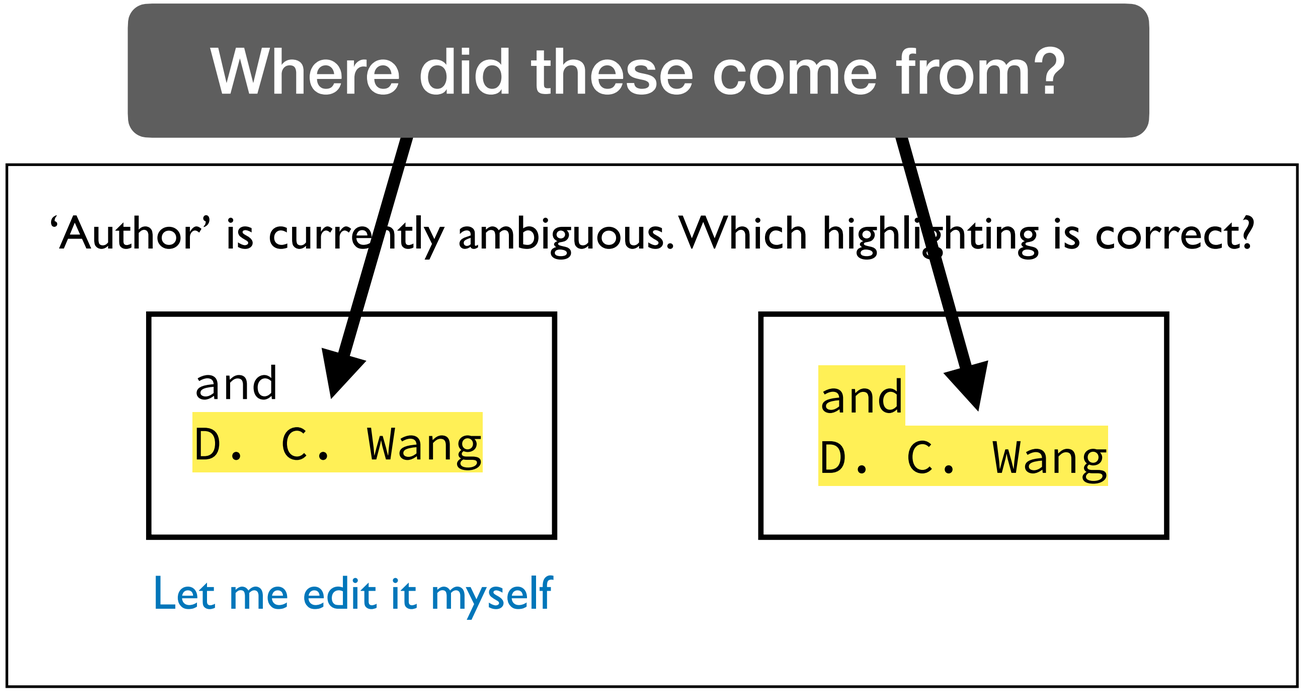


FlashProg (2015)

Mayer et al. "User Interaction Models for Disambiguation in Programming by Example". UIST 2015

Distinguishing Input

Synthesizer-Driven



First input that acts differently
on the top 2 programs

Slow convergence
Many interactions

Mayer et al. “User Interaction Models for Disambiguation in Programming by Example”. UIST 2015

Question Selection

Which distinguishing input?

Synthesizer-Driven

There is a better way!
Fewer interactions

Question Selection

Which distinguishing input?

Synthesizer-Driven

Bad News:

The Best Question is NP-Complete

Good News:

We can approximate it

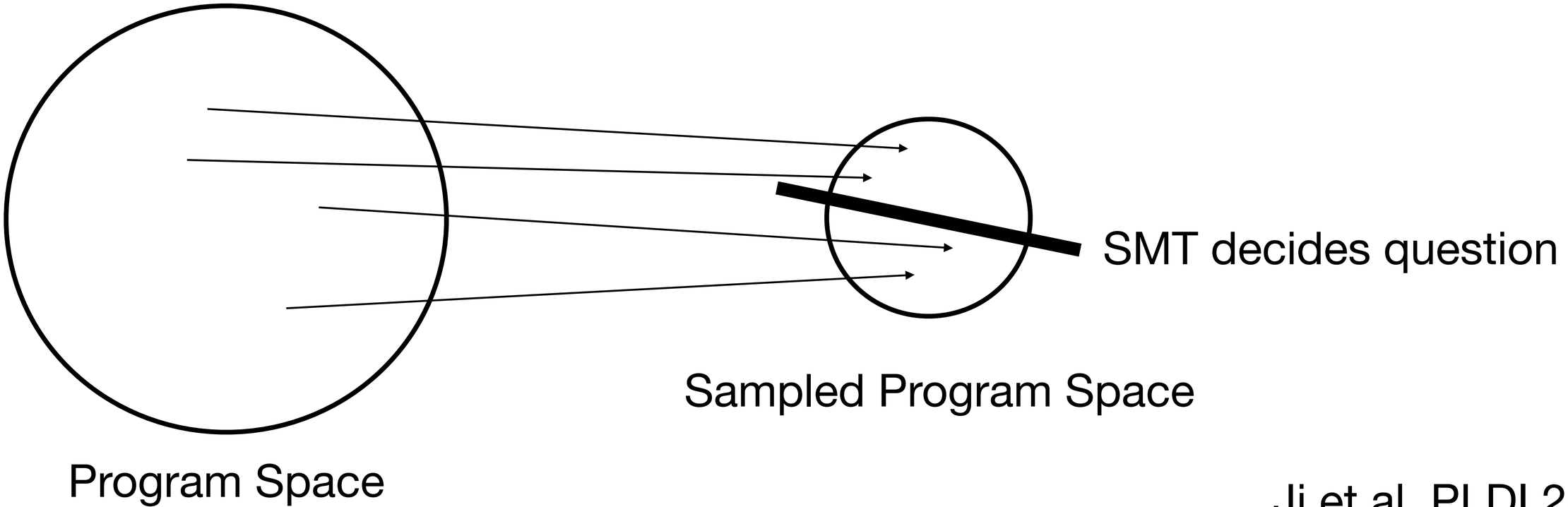
Ji et al. PLDI 2020

Question Selection

Which distinguishing input?

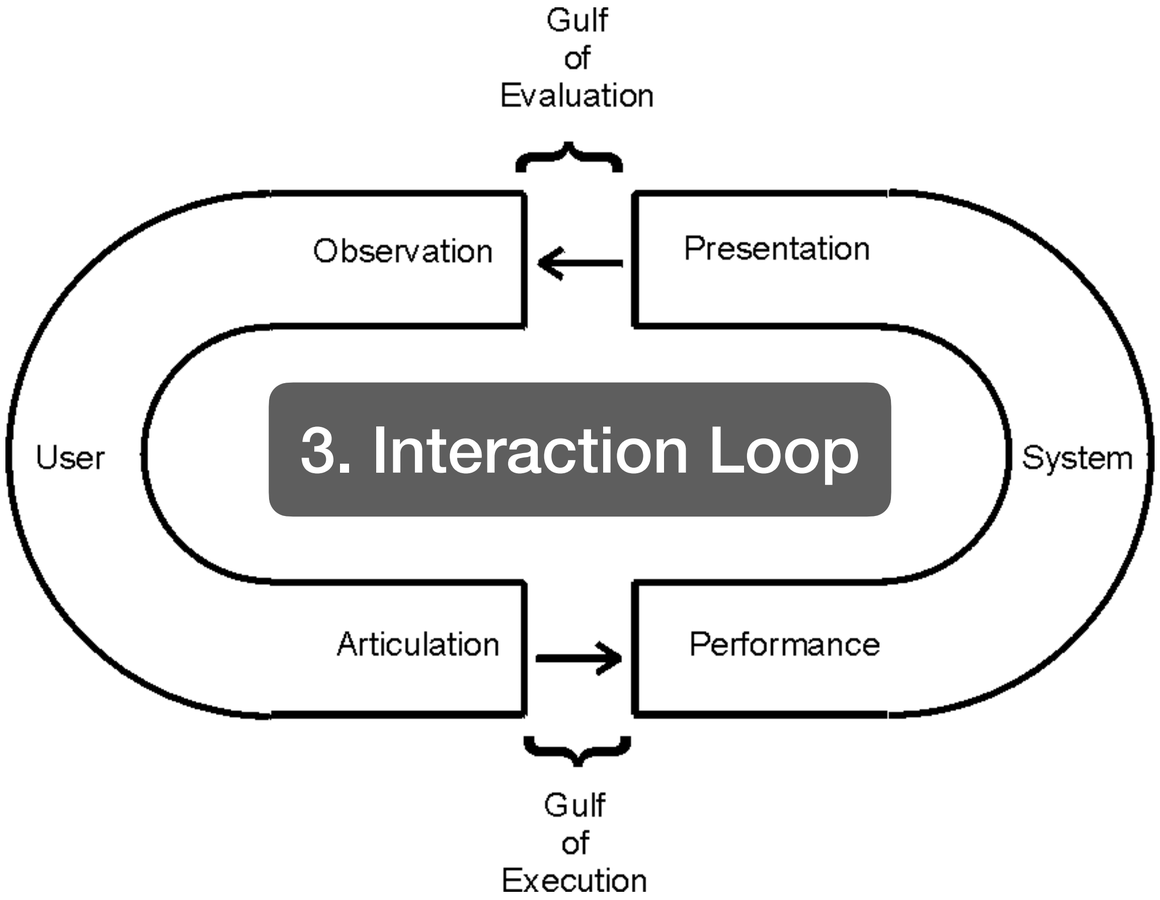
Synthesizer-Driven

Minimax over the user's input and a sampled search space.



Ji et al. PLDI 2020

Noisy Specification



User-Driven

Granular Interaction

Freeform Examples

Synthesizer-Driven

Disambiguating Example

Open Question

Noisy Specification

Users aren’t perfect

```
stutter “abc” = “aabbcc”
stutter “bc” = “bbcc”
stutter “c” = “bc”
stutter “” = “”
```

Most synthesizers

Timeout

Inconsistent specification

Open Question

Noisy Specification

Some systems handle noise

FlashFill	VSA Heuristic
RANSAC	Randomness
RobustFill	Probability
Bester	Best Effort

Open Question

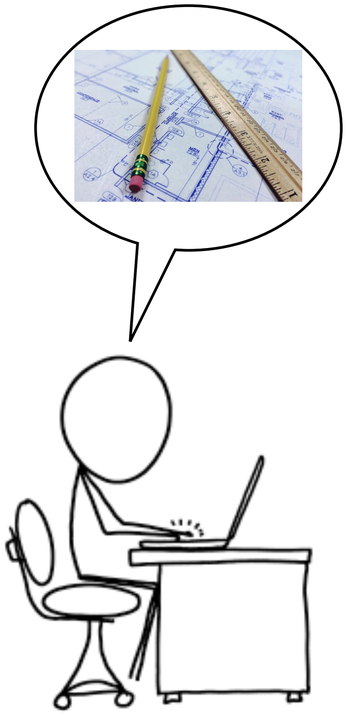
Noisy Specification

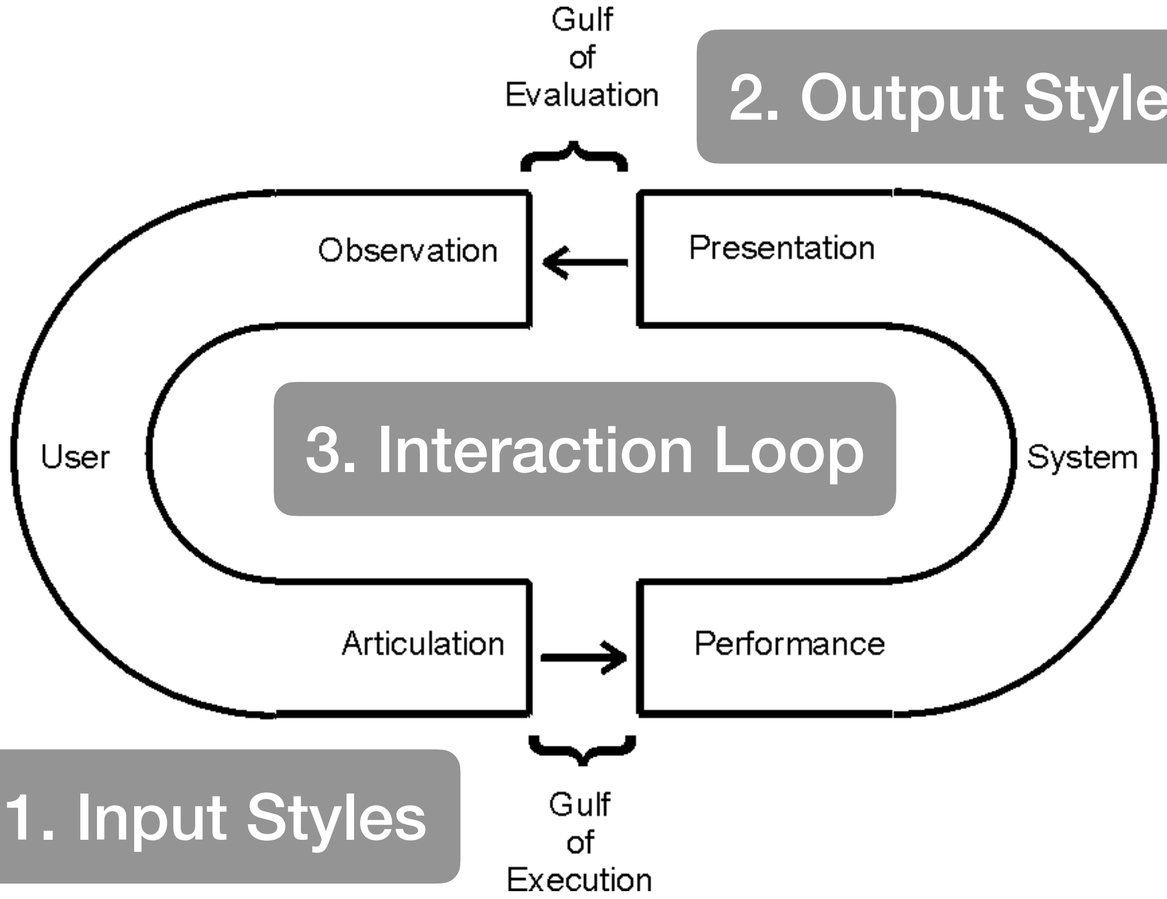
Some systems handle noise

FlashFill	VSA Heuristic
RANSAC	Randomness
RobustFill	Probability
Bester	Best Effort

None of these are general purpose synthesis

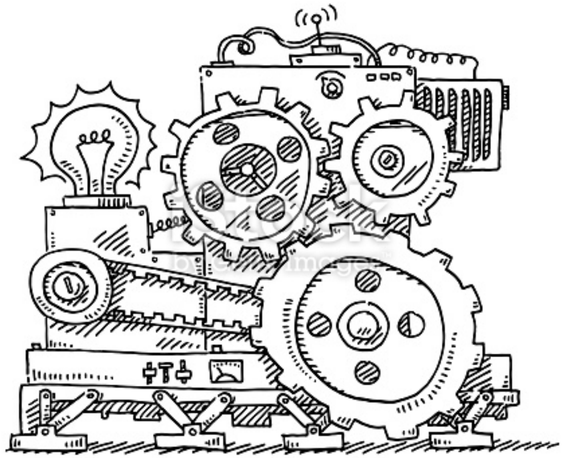
Can we synthesize with noisy input in general?





2. Output Styles

3. Interaction Loop



1. Input Styles

4. My Work

Hoogle+

5. Open Questions

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

Problem

Short code-snippet search in Haskell

API Discovery

```
dedup :: Eq a => [a] -> [a]
dedup [1,2,2,3,3,1] == [1, 2, 3, 1]
dedup xs = map head (group xs)
```

Input

Type | Type + Example | Example

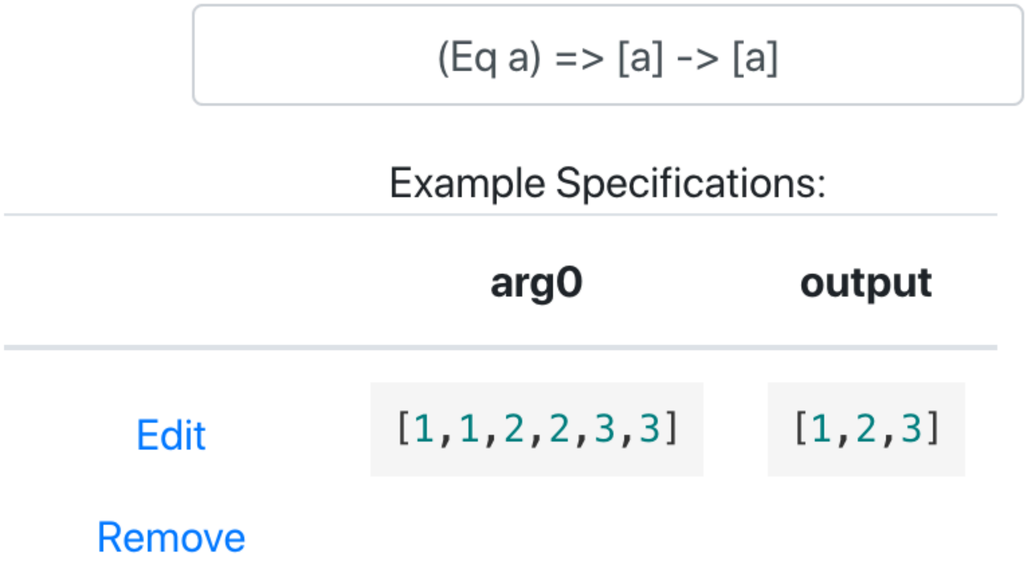
```
(Eq a) => [a] -> [a]
```

Input

Type

Type + Example

Example



Input

Type

Type + Example

Example

Example Specifications:

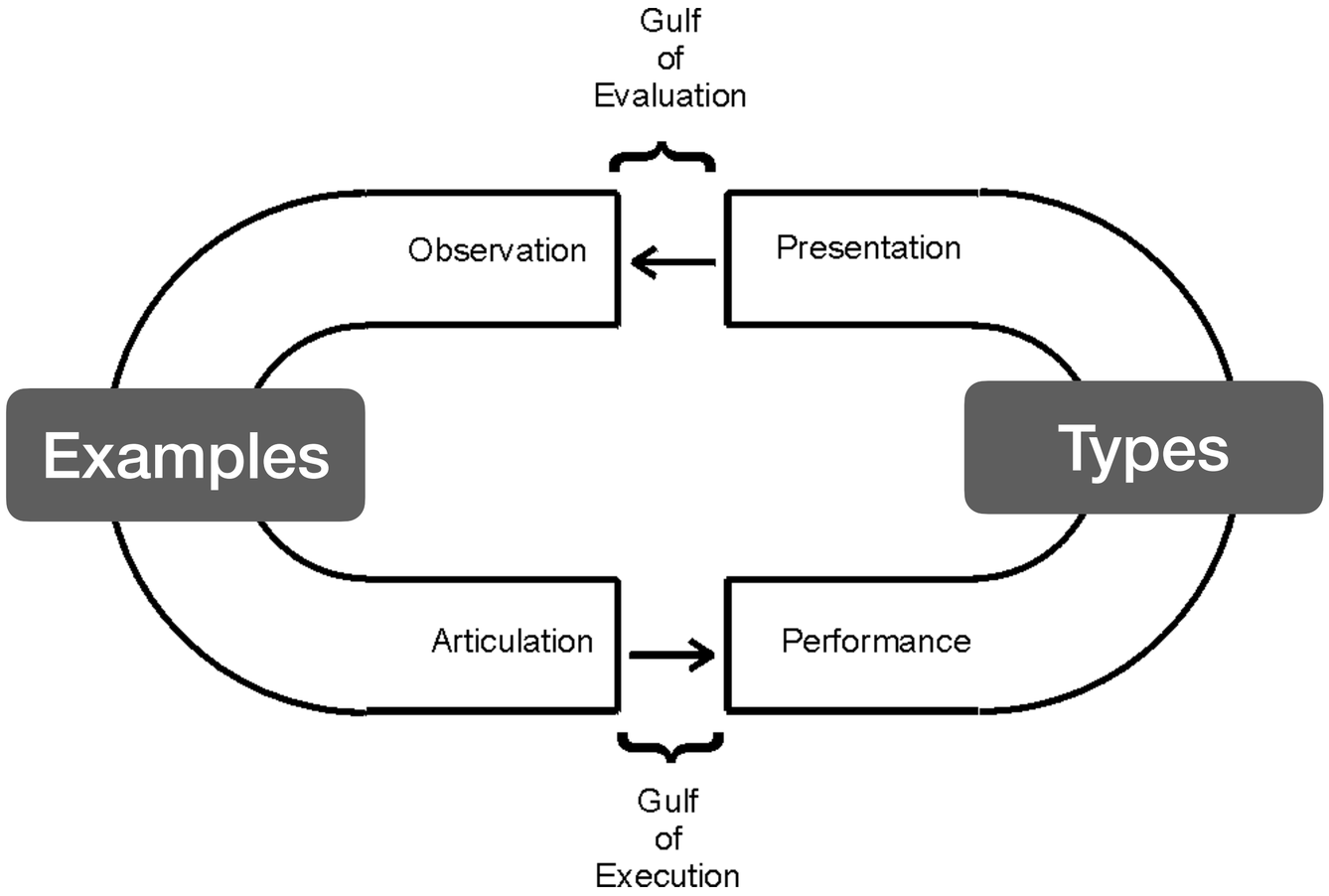
	arg0	output
Edit Remove	"aabaa"	"aba"
Edit Remove	[1,1,2,2,3,3]	[1,2,3]

Input - Examples in a typed world

Task: write a deduplication function

Example Specifications:

	arg0	output
Edit Remove	`"aabaa"`	`"aba"`
Edit Remove	`[1,1,2,2,3,3]`	`[1,2,3]`



Input - Examples in a typed world

Task: write a deduplication function

New Input Mode!

Example Specifications:

	arg0	output
Edit Remove	`"aabaa"`	`"aba"`
Edit Remove	`[1,1,2,2,3,3]`	`[1,2,3]`

Select the best type

1. `[t0] -> [t0]`
2. `(Eq t0) => [t0] -> [t0]`
3. `(Ord t0) => [t0] -> [t0]`

Output

1 \arg0 -> GHC.List.map GHC.List.head (Data.List.group arg0)

New usage	arg0	output
Edit Keep usage	"aabaa"	"aba"
Edit Keep usage	[1,1,2,2,3,3]	[1,2,3]
Edit Keep usage	[]	[]

More Examples

Result as code

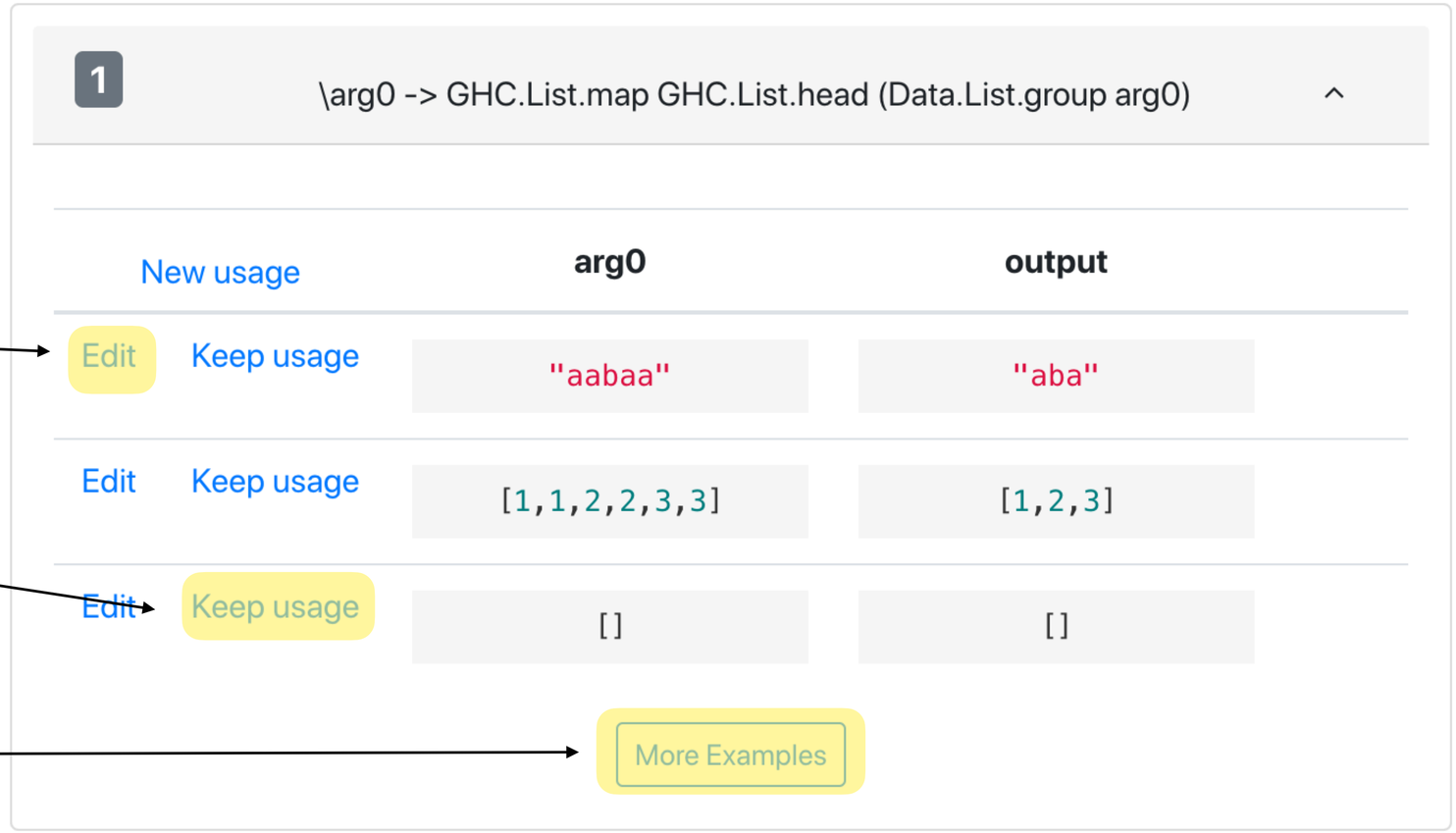
Extra Input-Output Examples

Interaction Loop

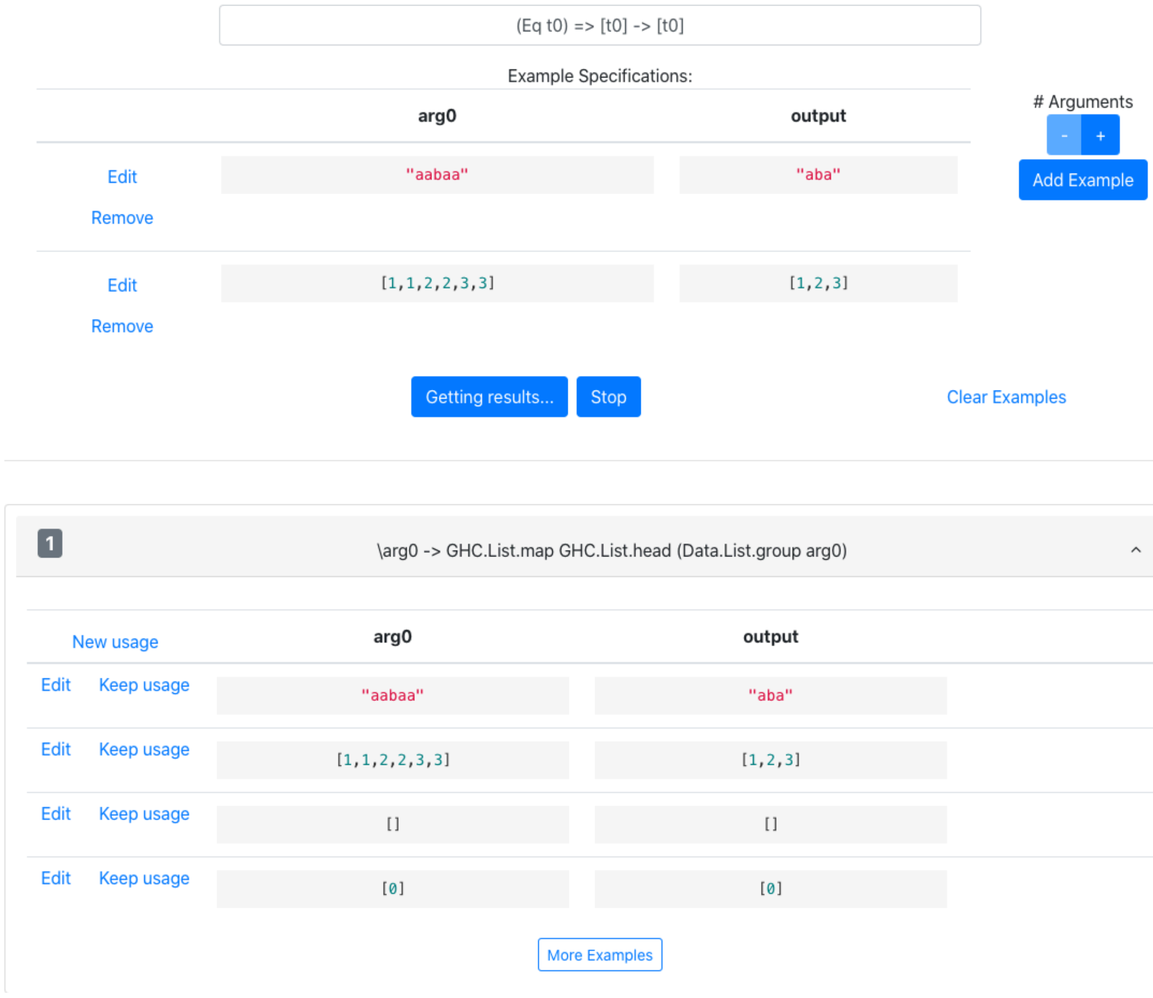
Live evaluation

Refine specification

More comprehension examples



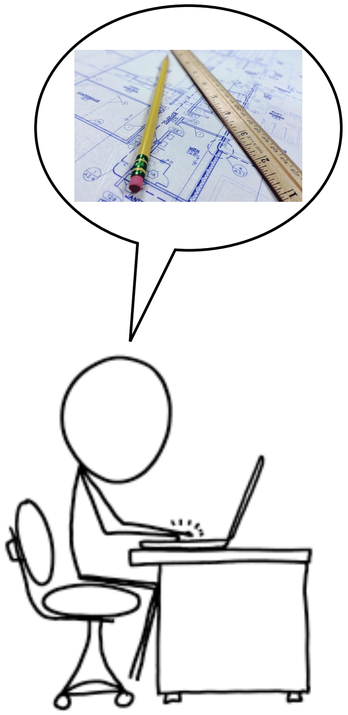
Hoogle+'s Future

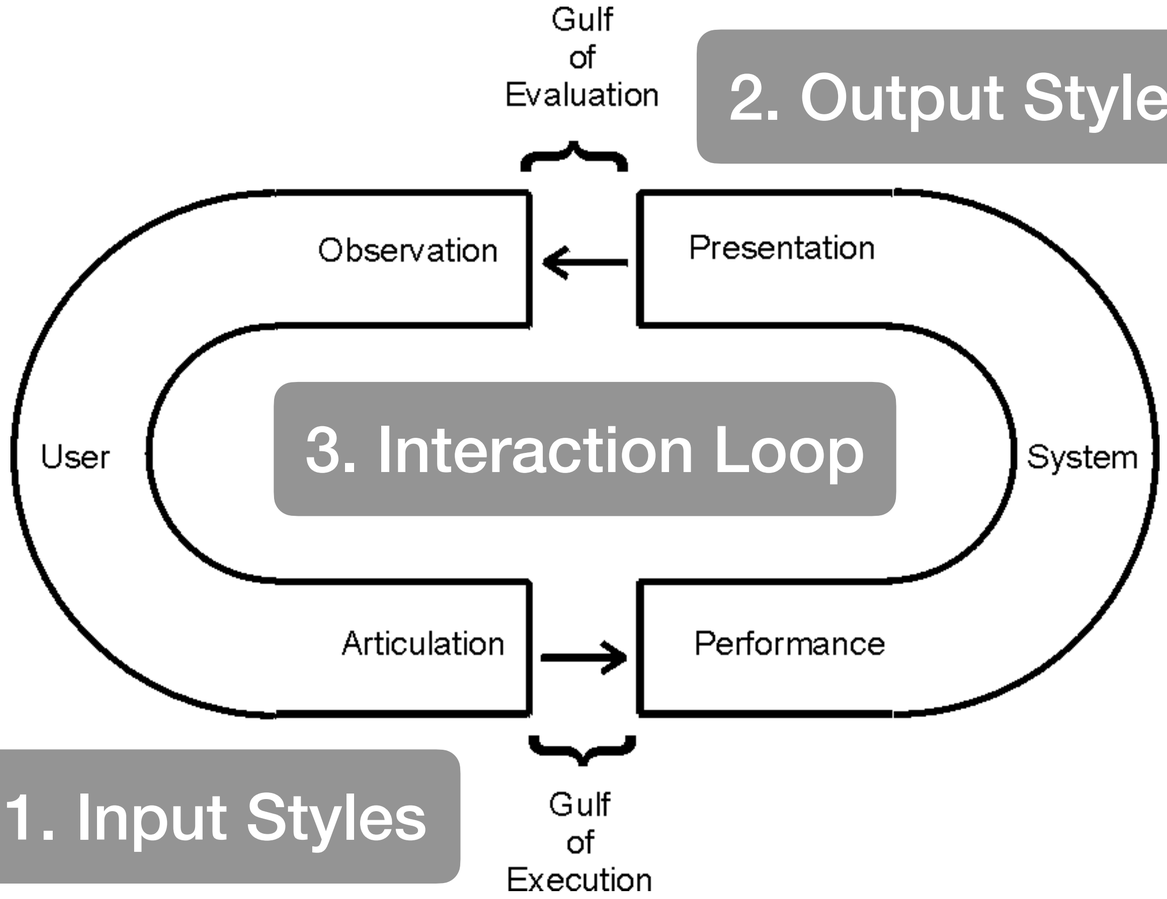


Different Differentiation

Comprehension Tools

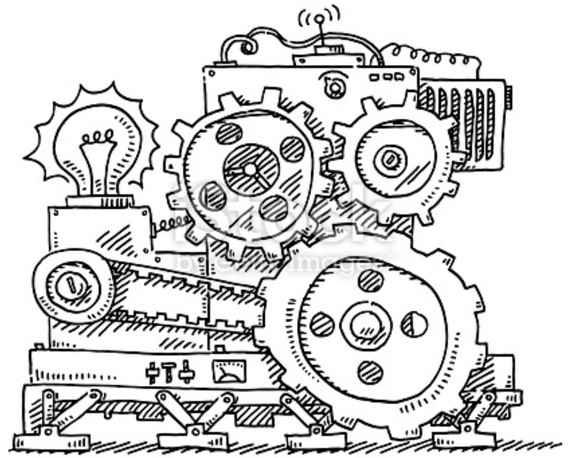
More input modalities





2. Output Styles

3. Interaction Loop



1. Input Styles

4. My Work

Hoogle+

5. Open Questions

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

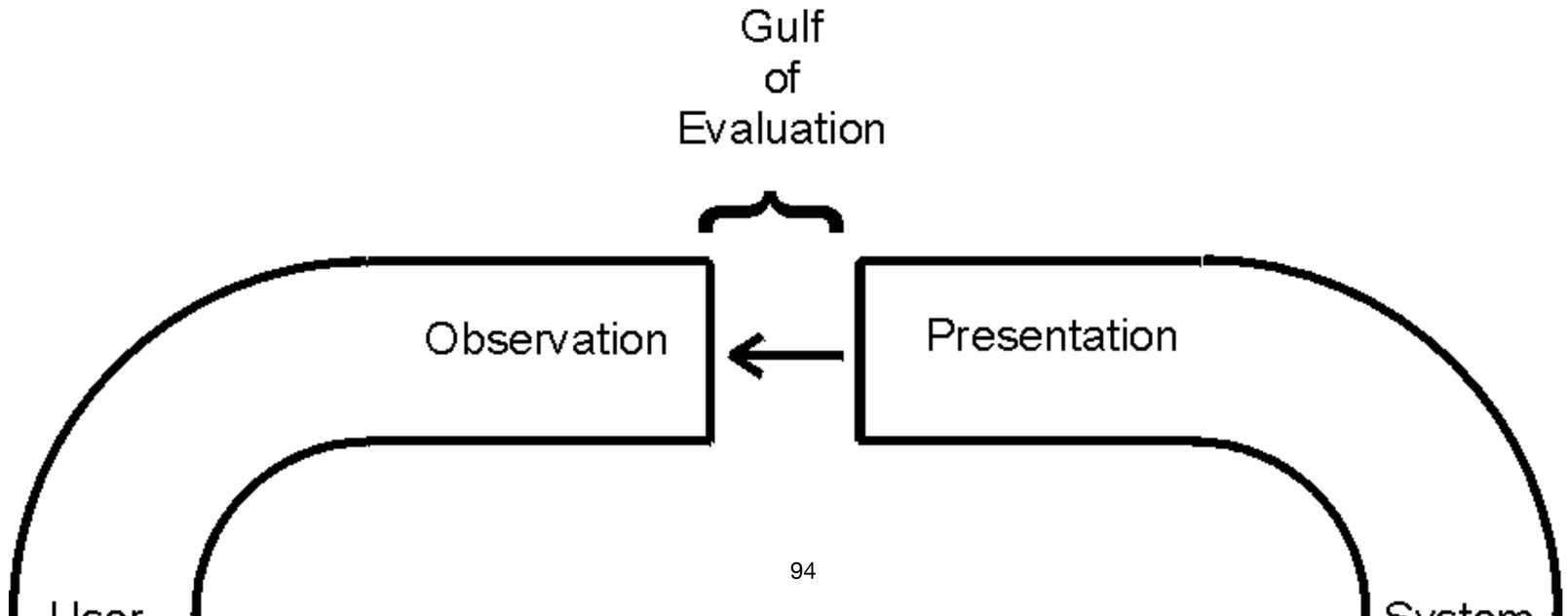
Understanding

Comprehension

Why does *this* program work for me?

Differentiation

Which program looks more promising?

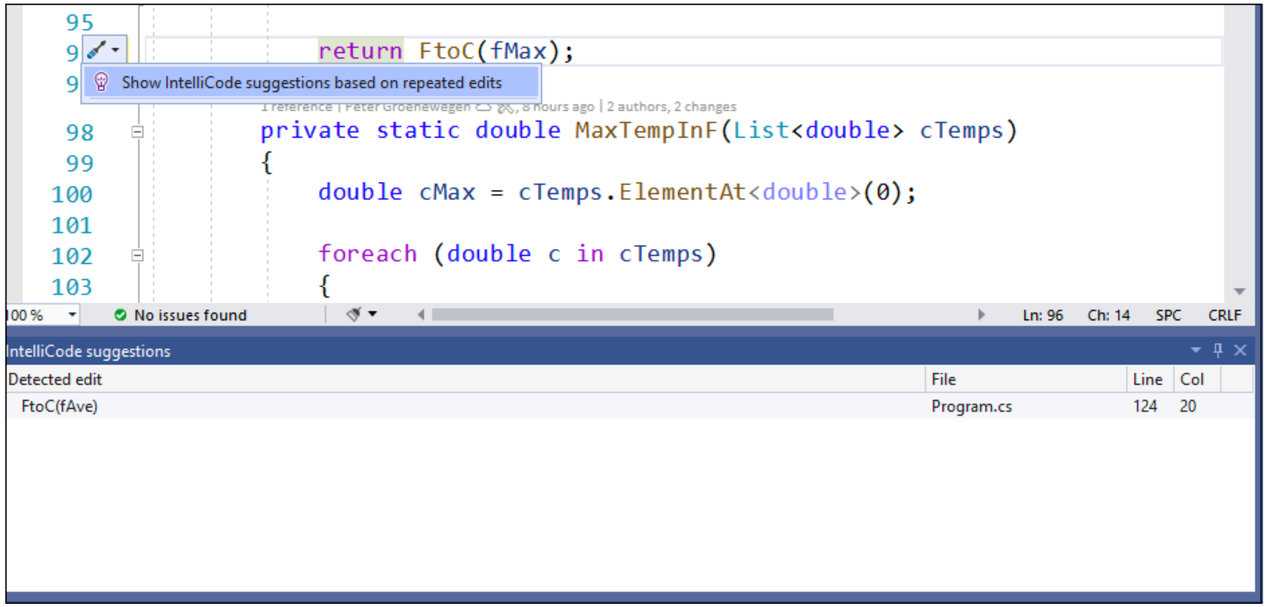


Tooling

Built into IDE

Developers don’t like to leave their IDEs

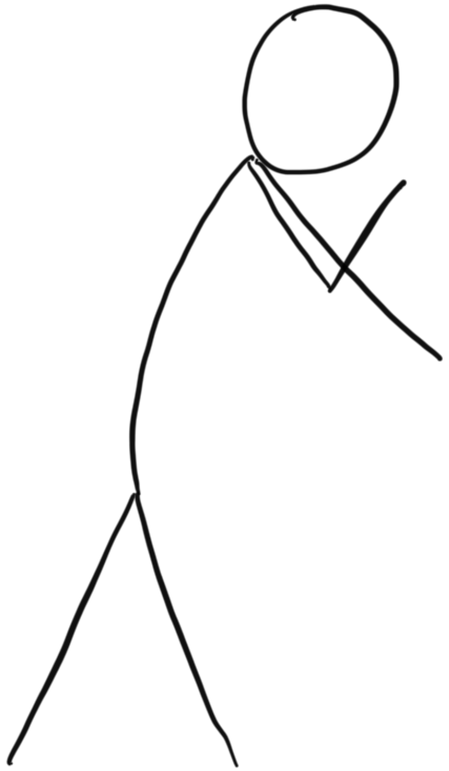
If you build it, they will use it

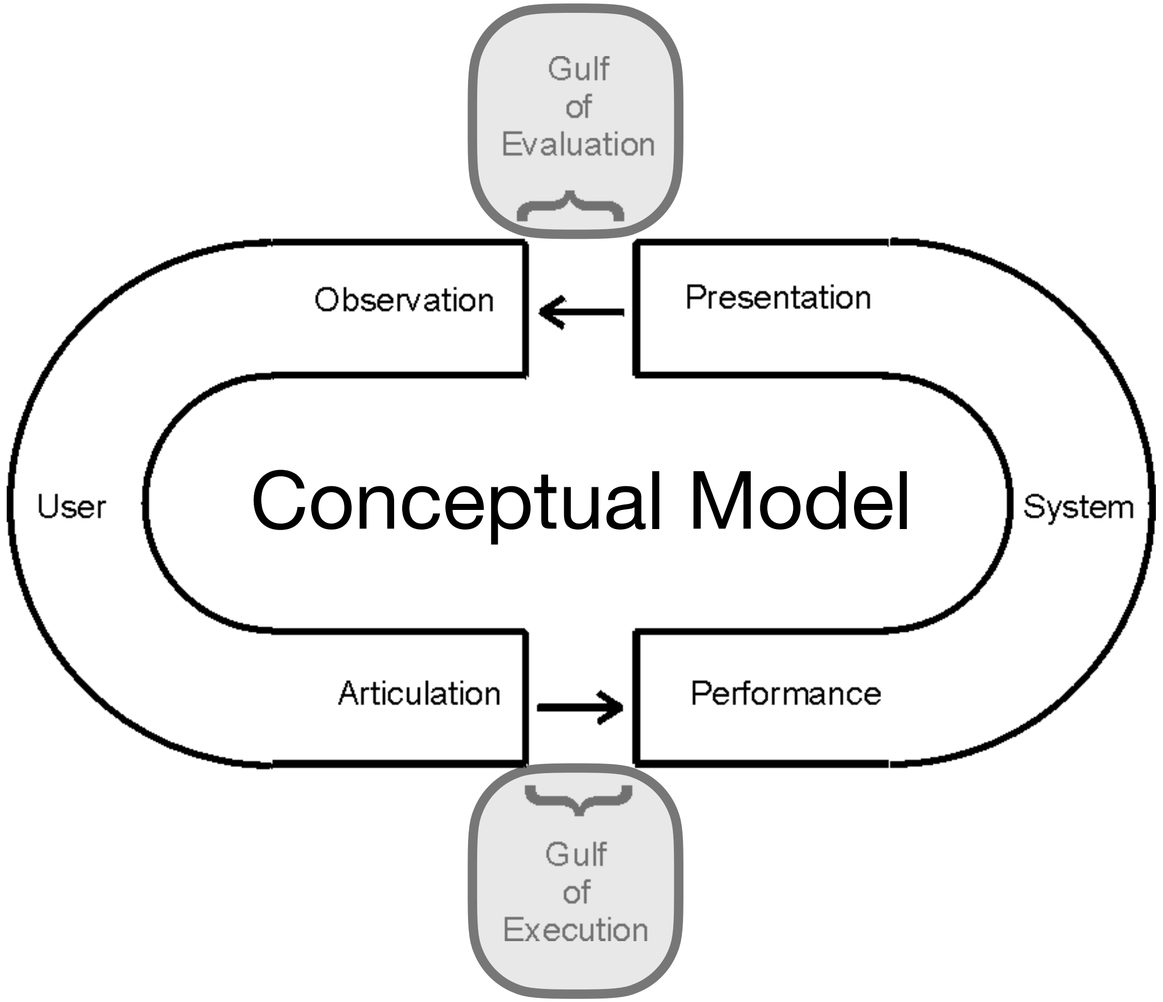


Blue-Pencil in IntelliCode from Microsoft

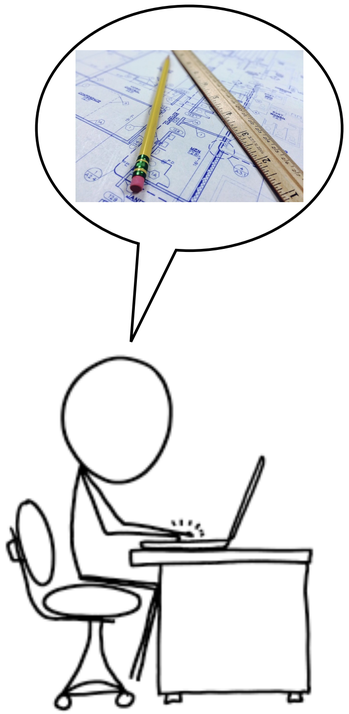
Trust

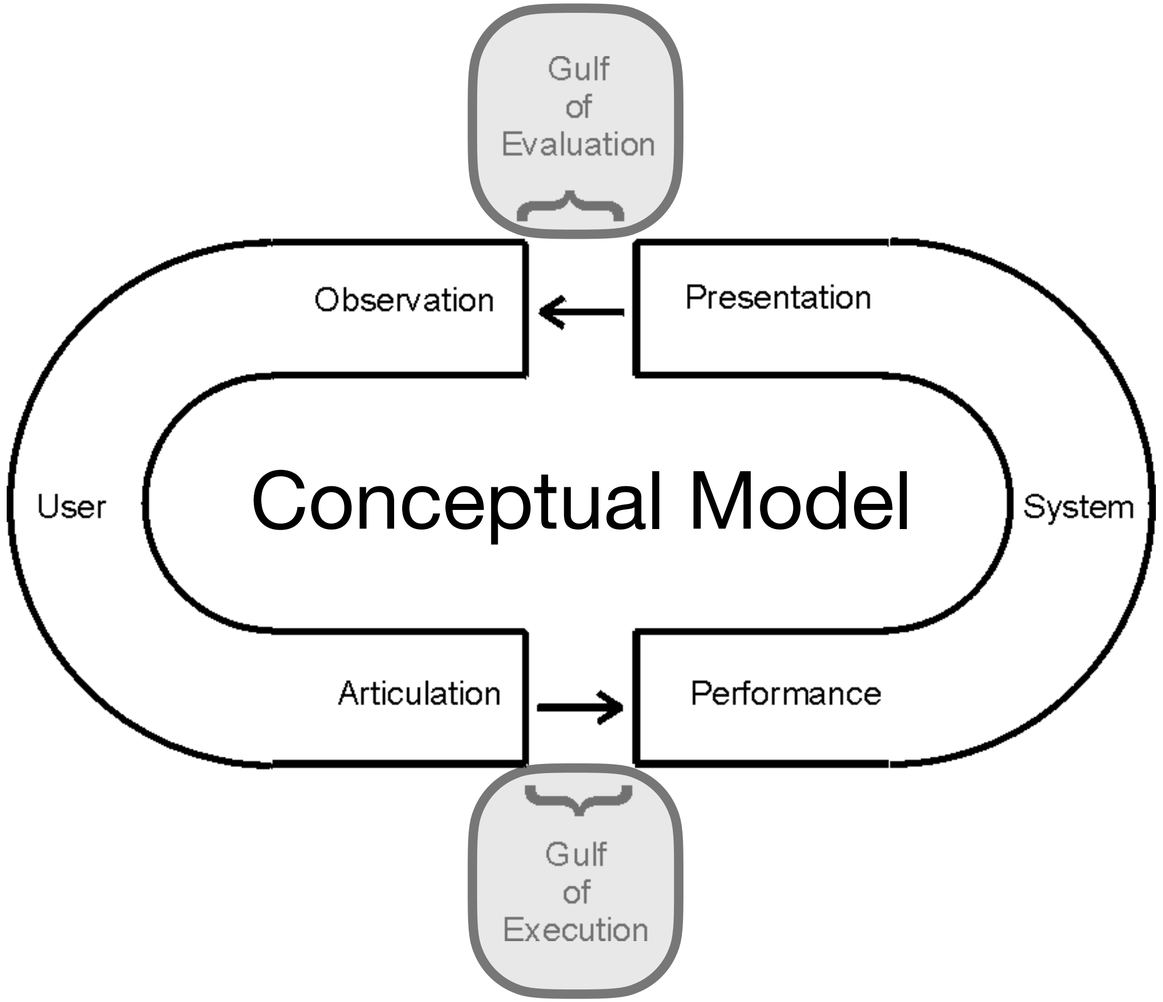
Trust

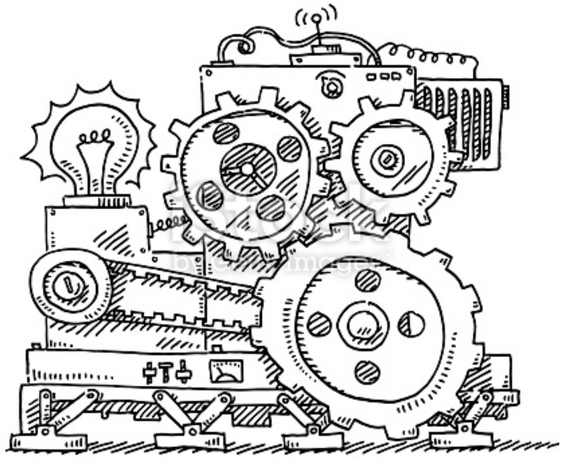


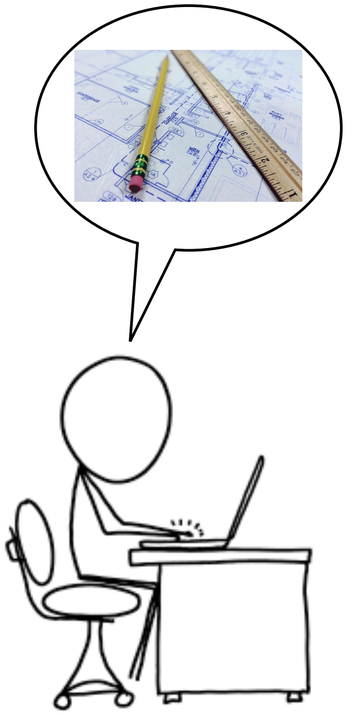

Gulf
of
Evaluation
Observation
Presentation
User
Conceptual Model
System
Articulation
Performance
Gulf
of
Execution

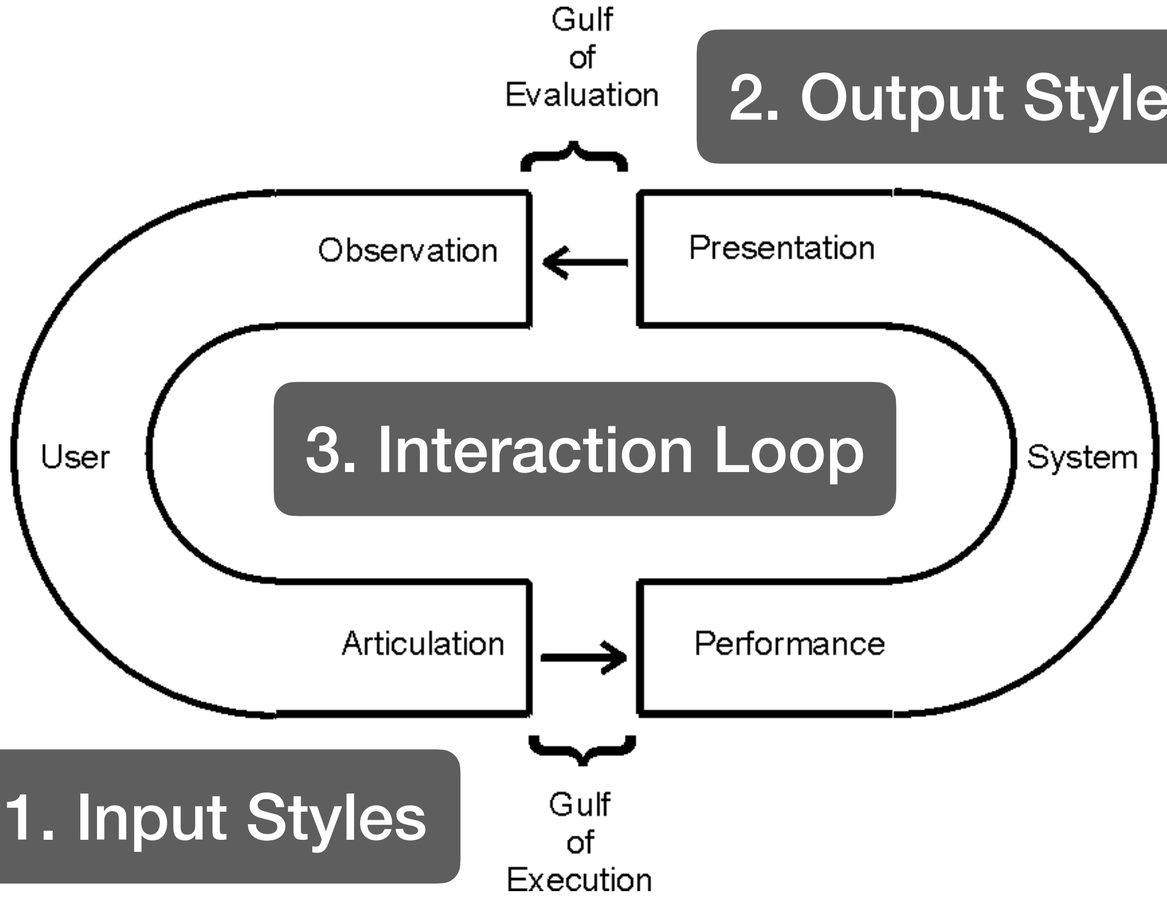
Trust




Gulf
of
Evaluation
Observation
Presentation
User
Conceptual Model
System
Articulation
Performance
Gulf
of
Execution

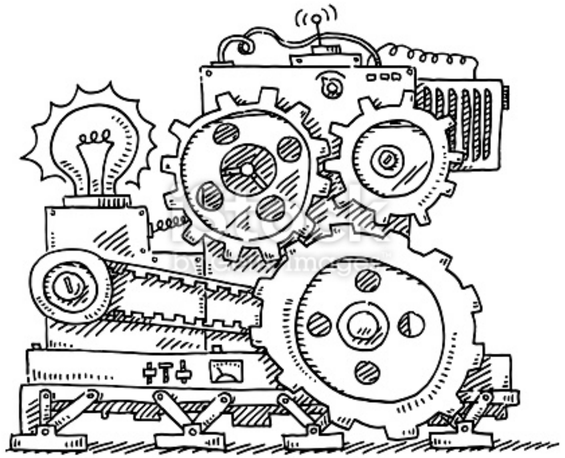






2. Output Styles

3. Interaction Loop



1. Input Styles

4. My Work

Hoogle+

5. Open Questions

http://cs4760.csl.mtu.edu/2016/lectures/normans-interaction-theory/

Thank You!