Today's themes & topics

- DSLs: Why, what, and how?
  When is a DSL appropriate?
  How can we tell if something is a DSL?
  How do we start implementing a DSL?
- Introducing fluency
- Terminology
  domain expert / user · implementor · domain analysis · internal DSL · external DSL · host language · fluency

Discussion: DSL benefits / drawbacks

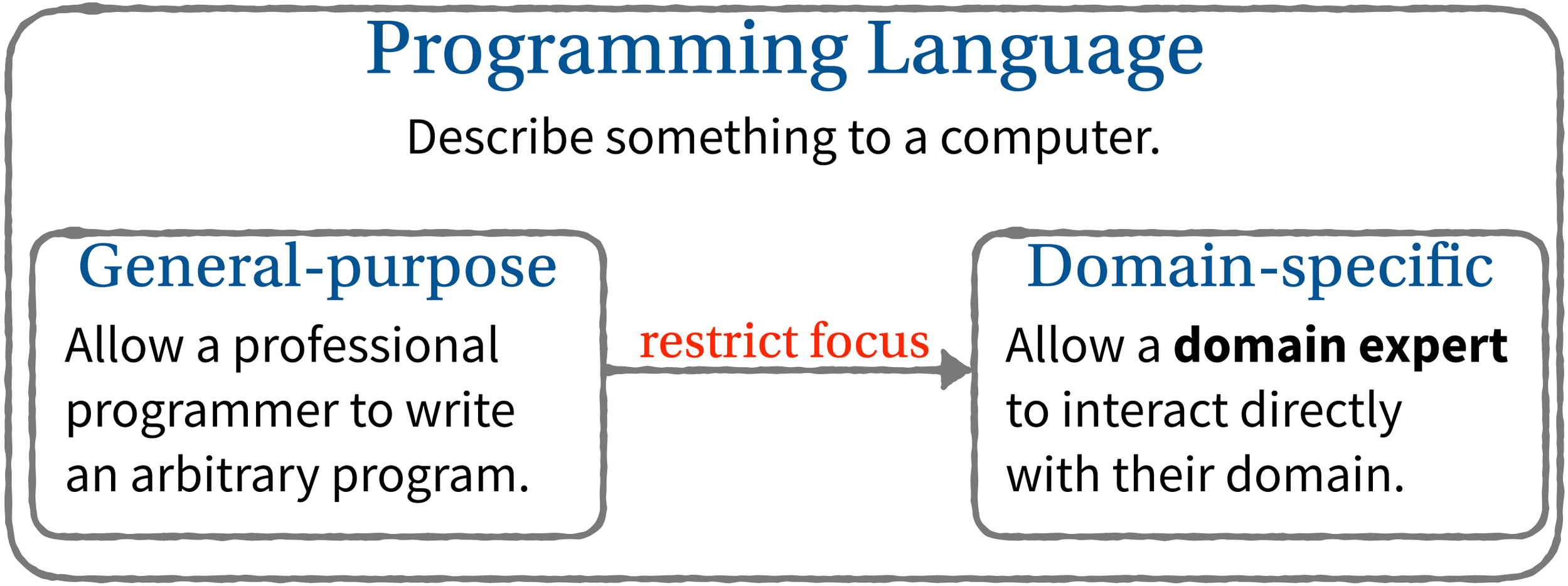
Benefits

- Communication w/ domain experts (use *their* language)
    - Especially when domain experts aren't experts in a PL.
- Add functionality to an existing PL (Internal DSL is an extension for host language)
    - (Even) when the domain experts are experts in a PL
- Limited expressiveness ⇒ more (machine) efficient

Drawbacks

- Limited expressiveness (only useful in its domain)
- Implementers pay a cost to make the DSL.
- Domain experts pay a cost ($, time for learning)
    - Tower of Babel

Is it a DSL?



We should have good answers for all these questions

1. Is it a programming language?
2. What is the focus? What does the *domain expert* describe?
3. What is easy, difficult, impossible in this language?

   (relative to a general-purpose programming language)

It’s a spectrum, not a binary

“GPPL-like” “DSL-like”

How precise are our answers to these questions?

1. Is it a programming language?
2. What is the focus? What does the *domain expert* describe?
3. What is easy, difficult, impossible in this language?

   (relative to a general-purpose programming language)

How do we implement a DSL?

1. Talk to the **domain experts** (a.k.a. the **users**).
2. Come up with a prototype.
3. Goto step 1.

Exercise: Image manipulation

- Warp
- Transform
- Mirror
- Crop
- Crop out a particular object in the image
- Brighter
- Dimmer
- More colorful
- Change hue
- Change tone
- Pixelate
- Sharpen
- Blur the background (to change focus)
- Selection a portion of a picture
- Add text to a picture

What tools does a GPPL give you?

- Loops
- Conditionals
- Variables
- Import / package
- Functions
- Tools that help you program
- Read inputs
- Write outputs

- Syntax: rules for correct programs
- Data structures: lists, arrays, etc.
- Complex calculation: built-in algorithms
- Mathematical operators

- Primitive Values (numbers)

Program

=

Data + Operations

Data structures
Types
Variables
Information
…

Algorithms
Functions, Methods,…
Loops, Conditionals,…
Behavior
…

Domain

=

Nouns + Verbs

Exercise: Image manipulation

- Warp
- Transform
- Mirror
- Crop
- Crop out a particular object in the image
- Brighter
- Dimmer
- More colorful
- Change hue
- Change tone
- Pixelate
- Sharpen
- Blur the background (to change focus)
- Select a portion of a picture
- Add text to a picture

How do we implement a DSL?

1. Talk to the **domain experts** (a.k.a. the **users**).
2. Come up with a prototype.
3. Goto step 1.

Initial prototype: a library in your GPPL

```
flipHorizontal(inputFilename, outputFilename)
flipVertical(inputFilename, outputFilename)
rotateLeft(inputFilename, outputFilename)
rotateRight(inputFilename, outputFilename)
grayScale(inputFilename, outputFilename)
```

bird.png

```
flipHorizontal(“bird.png”, “drib.png”)
```

drib.png

photo credit: Natasha Miller @ unsplash

```
def flipHorizontal(inputFilename, outputFilename):
    …
…
```

Picture library

```
# flip a picture horizontally, grayscale it, and rotate it left
Picture.flipHorizontal("image.png", "output0.png")
Picture.grayScale("output0.png", "output1.png")
Picture.rotateLeft("output1.png", "output2.png")
```

Python

```
void flipHorizontal(String inputFilename, String outputFilename) {
    …
}
…
```

Picture library

```
public static void main(String[] args) {
    // flip a picture horizontally, grayscale it, and rotate it left
    Picture.flipHorizontal("image.png", "output0.png");
    Picture.grayScale("output0.png", "output1.png");
    Picture.rotateLeft("output1.png", "output2.png");
}
```

Java

Second prototype: a library in your GPPL

```
loadImage(filename) => picture
flipHorizontal(picture) => picture
flipVertical(picture) => picture
rotateLeft(picture) => picture
rotateRight(picture) => picture
grayScale(picture) => picture
saveImage(picture, filename)
```

bird.png

```
saveImage(flipHorizontal(loadImage("bird.png")), "drib.png")
```

drib.png

photo credit: Natasha Miller @ unsplash

Implementing DSLs: terminology

user: a person who writes programs in the DSL

implementer: a person who makes the DSL for the user(s)

implementation PL: what the implementer uses to make the DSL

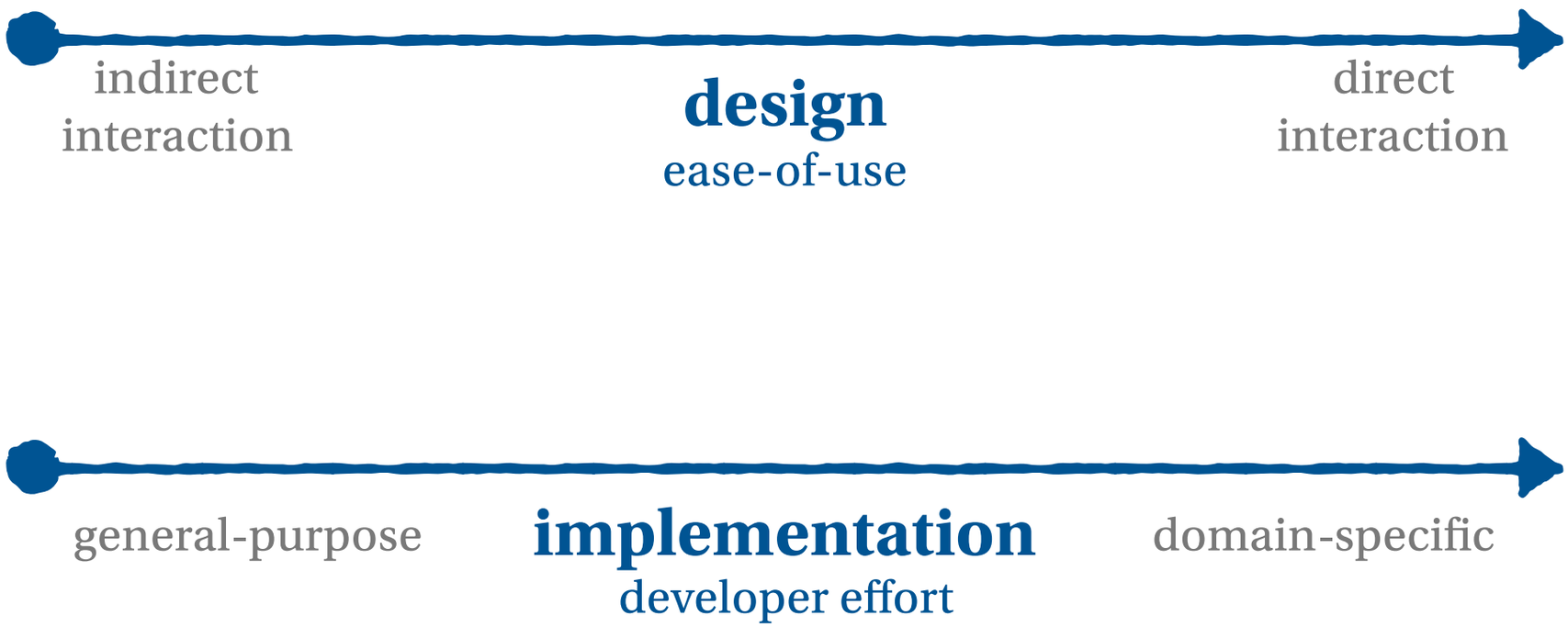
internal DSL: a DSL where each DSL program is *also* a valid program in the implementation PL (called the **host PL**)

external DSL: a DSL where a DSL program might not be a valid program in any other PL.

fluency: the degree to which a DSL's nouns/verbs fit together. Can we build something big from smaller pieces?

How do we implement a DSL?

1. Talk to the **domain experts** (a.k.a. the **users**).
2. Come up with a prototype.
3. Goto step 1.



Implementation techniques