Programming across Paradigms

@AnjanaVakil

GOTO Chicago 2017
hi, I’m @AnjanaVakil!

Über Research

The Recurse Center

mozilla

OUTREACHY

MOZILLA TECH SPEAKERS
The Paradigms of Programming

Robert W. Floyd
Stanford University

Today I want to talk about the paradigms of programming, how they affect our success as designers of computer programs, how they should be taught, and how they should be embodied in our programming languages.

A familiar example of a paradigm of programming is the technique of structured programming, which appears to be the dominant paradigm in most current treatments of programming methodology. Structured programming, as formulated by Dijkstra [6], Wirth [27, 29], and Parnas [21], among others, consists of two phases.

In the first phase, that of top-down design, or stepwise refinement, the problem is decomposed into a very small number of simpler subproblems. In programming the solution of simultaneous linear equations, say, the first level of decomposition would be into a stage of triangularizing the equations and a following stage of back-substitution in the triangularized system. This gradual decomposition is continued until the subproblems that arise are simple enough to cope with directly. In the simultaneous equation example, the back substitution process would be further decomposed as a backwards iteration of a process which finds and stores the value of the $i$th variable from the $i$th equation. Yet further decomposition would yield a fully detailed algorithm.
“I believe the best chance we have to improve the general practice of programming is to attend to our paradigms.”

Robert W. Floyd

“The paradigms of programming.”, 1979. p. 456
what is a paradigm?
a paradigm is a **worldview**
a paradigm is a model
a paradigm enables progress
“In learning a paradigm the scientist acquires theory, methods, and standards together, usually in an inextricable mixture.”

Thomas S. Kuhn

theory

what entities make up the universe
how they behave and interact
methods & standards

which problems are worth solving
which solutions are legitimate
“All models are wrong”

George E. P. Box

what are some major paradigms?
imperative programming

follow my commands

in the order I give them

remember state
imperative programming
21    /* See comments at XXXROUNDUP below. Returns -1 on overflow. */
22    static int
23    fancy_roundup(int n)
24    {
25        /* Round up to the closest power of 2 \geq n. */
26        int result = 256;
27        assert(n > 128);
28        while (result < n) {
29            result <<= 1;
30            if (result <= 0)
31                return -1;
32        }
33        return result;
34    }
keep your state to yourself
receive my messages
respond as you see fit

object-oriented programming
object-oriented programming
# Structured result objects for string data

```python
class DefragResult(_DefragResultBase, _ResultMixinStr):
    __slots__ = ()

def geturl(self):
    if self.fragment:
        return self.url + '#' + self.fragment
    else:
        return self.url
```
functional programming

mutable state is dangerous

pure functions are safe

data goes in data comes out
functional programming
(define map
  (lambda (f xs)
    (if (null? xs)
        '()
        (cons (f (car xs)) (map f (cdr xs))))))
declarative programming

does not care how

these are the
facts

this is what I
want

I don’t care how
you do it
declarative programming
```sql
SELECT isbn, title, price, price * 0.06 AS sales_tax
FROM Book
WHERE price > 100.00
ORDER BY title;
```
parent_child(juan, ana).
parent_child(kim, ana).
parent_child(kim, mai).

sibling(X,Y) :- parent_child(Z,X), parent_child(Z,Y).

?- sibling(ana, mai).
Yes
what do they have in common?
shared mutable state
shared mutable state
shared mutable state
shared mutable state
“I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea. The big idea is "messaging".”

Alan Kay

Message to Smalltalk/Squeak mailing list, 1998
thing.do(some,stuff)
recipient \rightarrow message

\texttt{thing}.\texttt{do}(\texttt{some}, \texttt{stuff})

method name \quad arguments
Ruby

def some(stuff)
  thing.send(:do, some, stuff)
end
import string

class Friend:
    def __init__(self, friends):
        self.friends = friends
    def is_friend_of(self, name):
        return name in self.friends

buddy = Friend(['alan', 'alonzo'])
buddy.is_friend_of('guy')  # False
buddy.is_friend_of('guy')
buddy.is_friend_of('guy')

buddy.send('is_friend_of', 'guy')
buddy.is_friend_of('guy')
buddy.send('is_friend_of', 'guy')
buddy('is_friend_of', 'guy')
def Friend(friend_names):
    def is_my_friend(name):
        return name in friend_names
    def instance(method, *args):
        if method == 'is_friend_of':
            return is_my_friend(*args)
        return instance

buddy = Friend(['alan', 'alonzo'])
buddy('is_friend_of', 'guy')  # False
class Friend:
    def __init__(self, friends):
        self.friends = friends
    def is_friend_of(self, name):
        return name in self.friends

buddy = Friend(['alan', 'alonzo'])
buddy.is_friend_of('guy')  # False
which paradigm is the best?
“All models are wrong...”

George E. P. Box

“All models are wrong but some are useful”

George E. P. Box

Each paradigm supports a set of concepts that makes it the best for a certain kind of problem.

Peter Van Roy

“Is the model true?”

“Is the model illuminating and useful?”

George E. P. Box

what can a paradigm teach me?
be explicit
understand
implementation
# global.py
for i in range(10**8):
    i

time: 9.185s

# in_fn.py

def run_loop():
    for i in range(10**8):
        i
    run_loop()

time: 5.738s

```python
# global.py
for i in range(10**8):
    i

time: 9.185s

# in_fn.py

def run_loop():
    for i in range(10**8):
        i
    run_loop()

time: 5.738s

1 # in_fn.py
2 def run_loop():
3     for i in range(10**8):
4         i
5     run_loop()
```

be abstract
understand
domain
Embedded DSL in Java

cal = new Calendar();
cal.event("GOTO Chicago")
  .on(2017, 5, 2)
  .from("09:00")
  .to("17:00")
  .at("Swissôtel");

Adapted from M. Fowler & R. Parsons, *Domain Specific Languages*, 2011, p. 345.
encapsulate
communicate
Context-aware API in F#

module MyApi =
    let fnA dep1 dep2 dep3 arg1 = doAWith dep1 dep2 dep3 arg1
    let fnB dep1 dep2 dep3 arg2 = doBWith dep1 dep2 dep3 arg2

    type MyParametricApi(dep1, dep2, dep3) =
        member __.FnA arg1 = doAWith dep1 dep2 dep3 arg1
        member __.FnB arg2 = doBWith dep1 dep2 dep3 arg2

specialize
transform data
get unreconciled dispensations

get settled deposits (that aren't all used)

oldest first

for each dispensation

try to match with a deposit

no deposit available

mark deposit used

mark donation reconciled

more?

commit

Adapted from J. Kerr, “Why Functional Matters: Your white board will never be the same”, 2012.
Adapted from J. Kerr, “Why Functional Matters: Your white board will never be the same”, 2012.
no paradigm is best absolutely
each is best for a certain case
“If the advancement of the general art of programming requires the continuing invention and elaboration of paradigms, ...”

Robert W. Floyd

“The paradigms of programming.”, 1979. p. 456
“If the advancement of the general art of programming requires the continuing invention and elaboration of paradigms, advancement of the art of the individual programmer requires that [the] repertory of paradigms.”

Robert W. Floyd

“The paradigms of programming.”, 1979. p. 456
learn new paradigms
try multi-paradigm languages
what’s the point?
paradigms enable programming
paradigms define programming
don’t fight your paradigm, embrace it
be open to shift
attend to your paradigms
David Albert, Darius Bacon, Julia Evans & the Recurse Center

GOTO Chicago organizers

thank you!

@AnjanaVakil


Fowler, Martin, with Parsons, Rebecca. (2011). *Domain Specific Languages*. Addison-Wesley.


