

Polly *want a* Message

Sandi Metz

Path to Pain



Design Stamina Hypothesis

– *Martin Fowler*



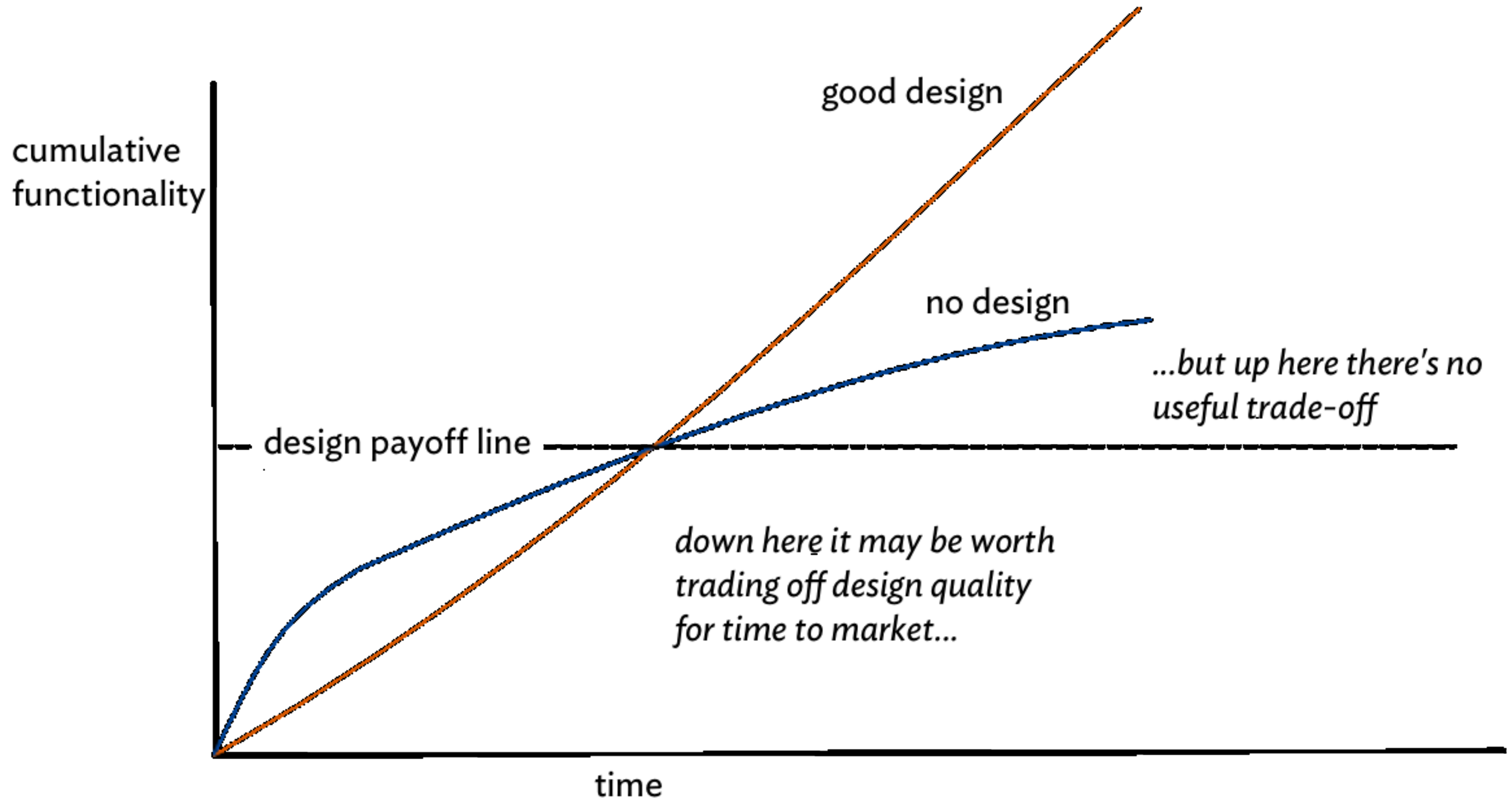
Design Stamina Hypothesis

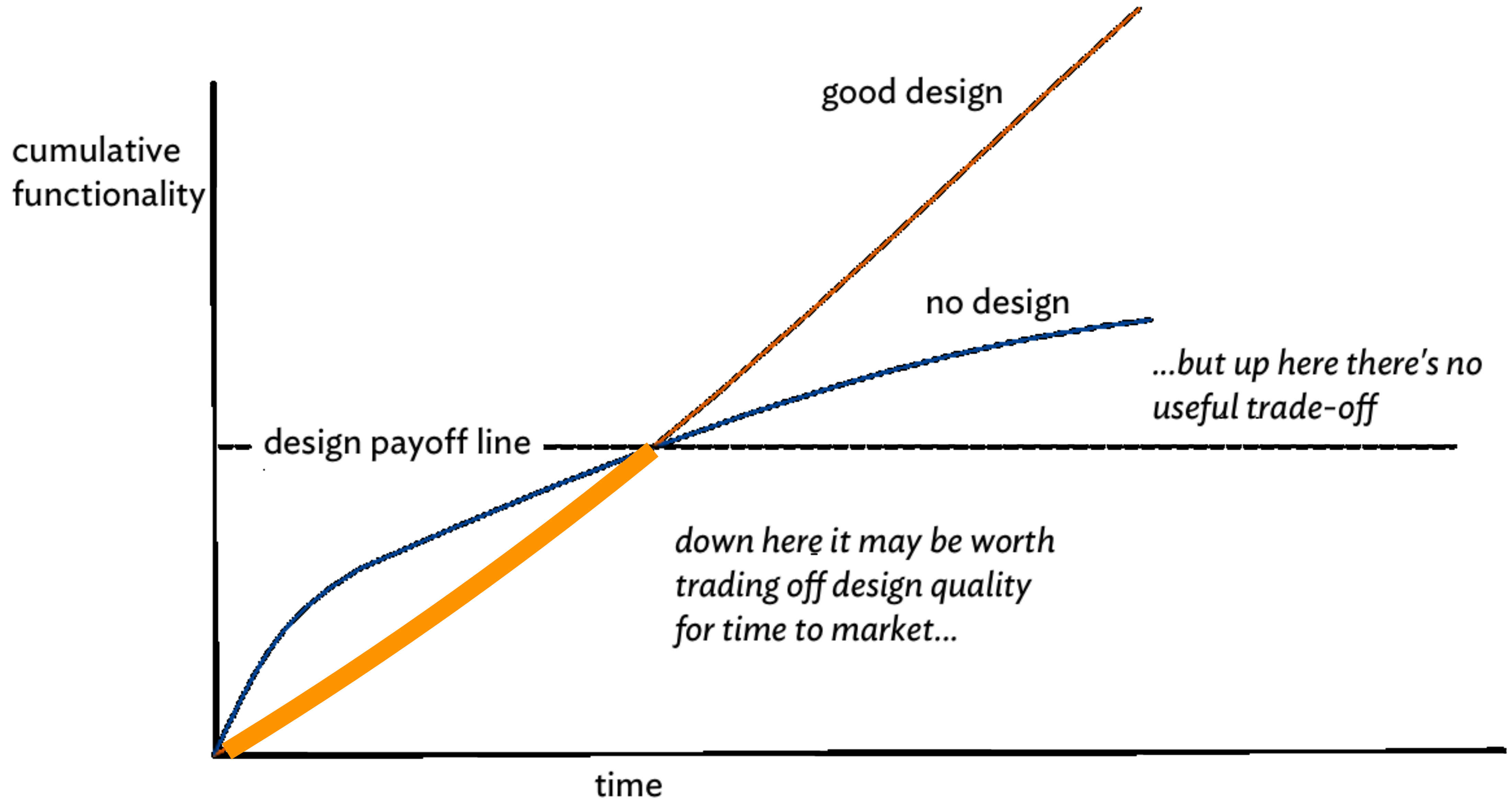
– *Martin Fowler*

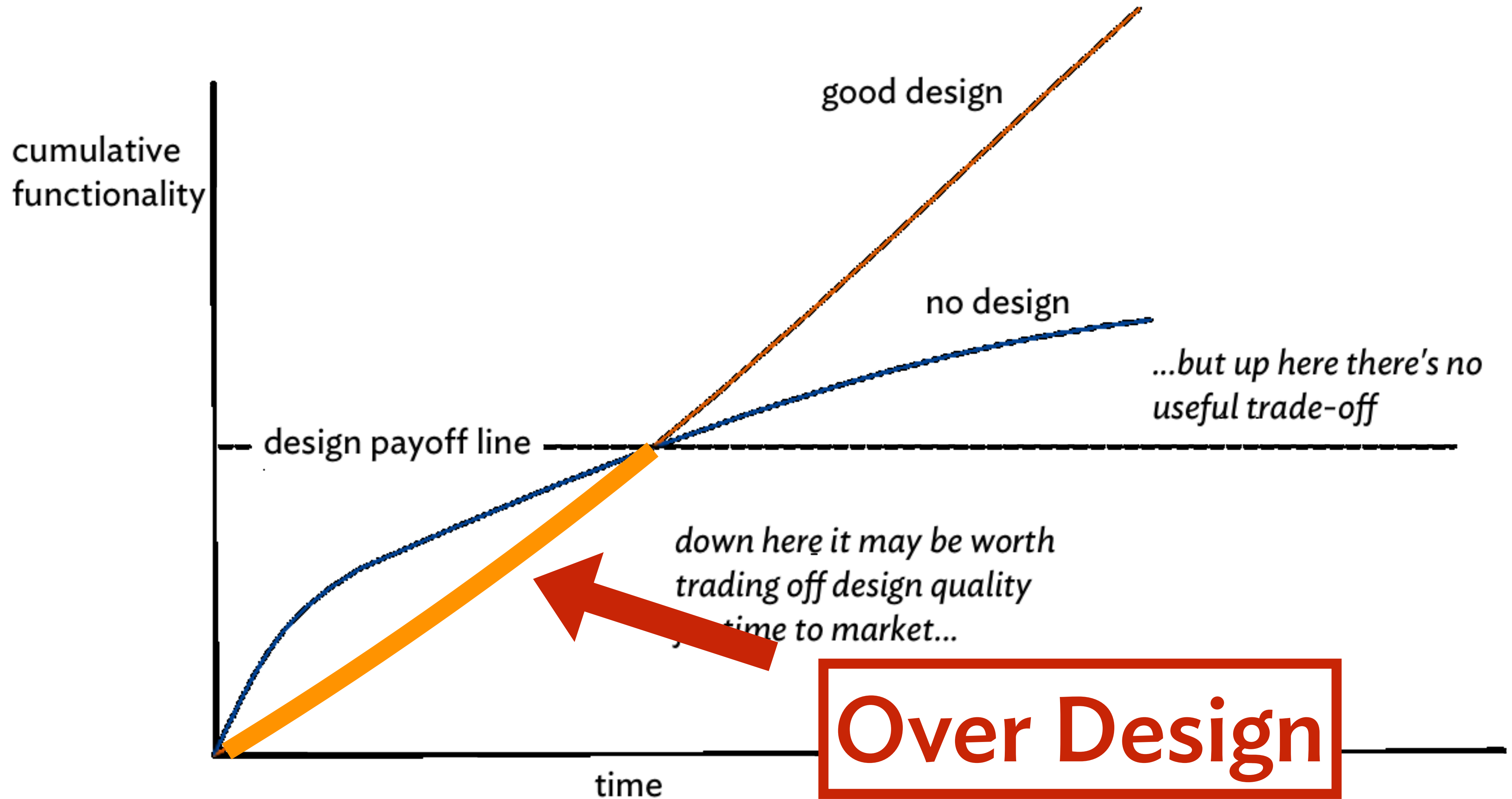


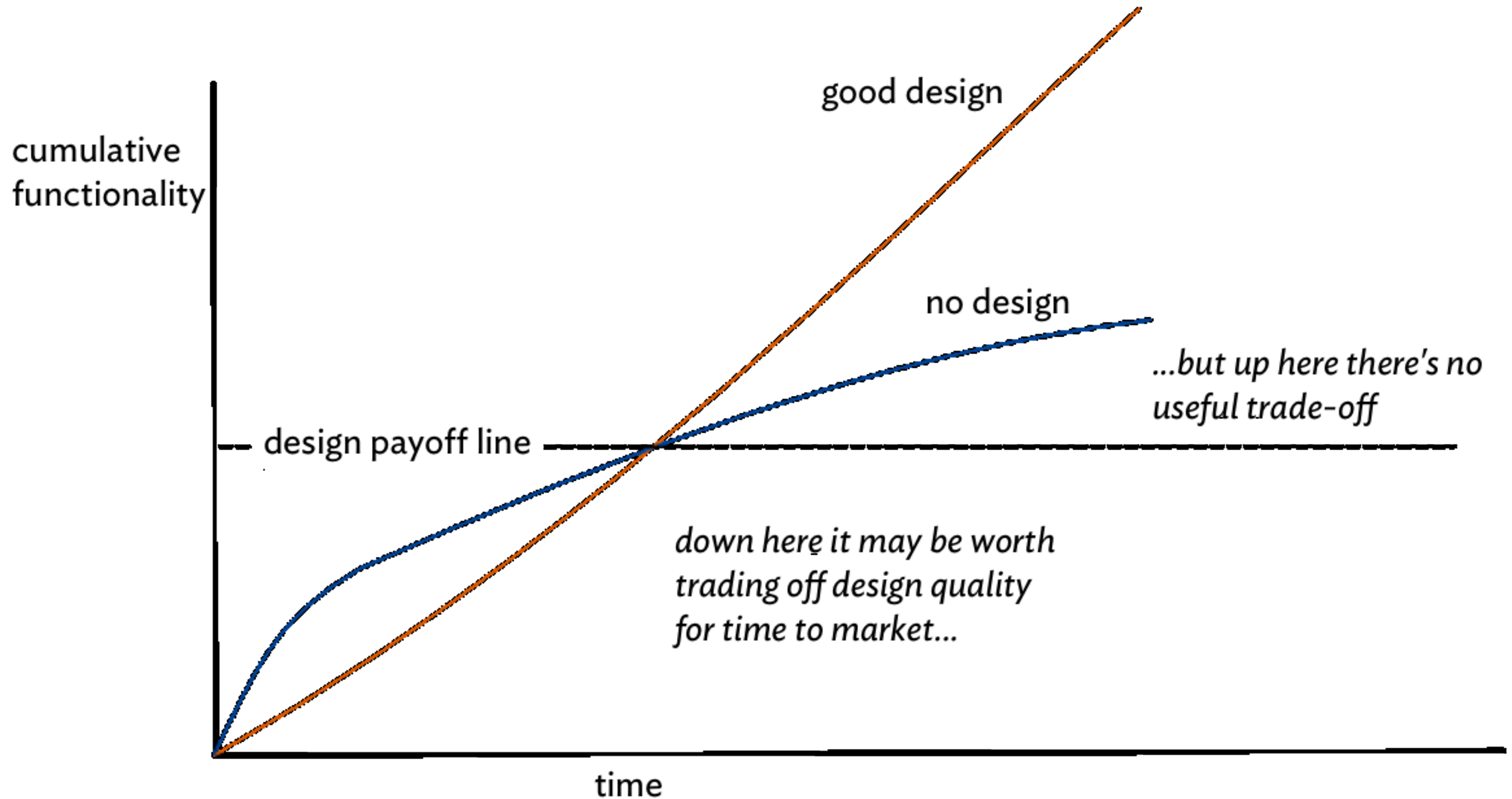
#1

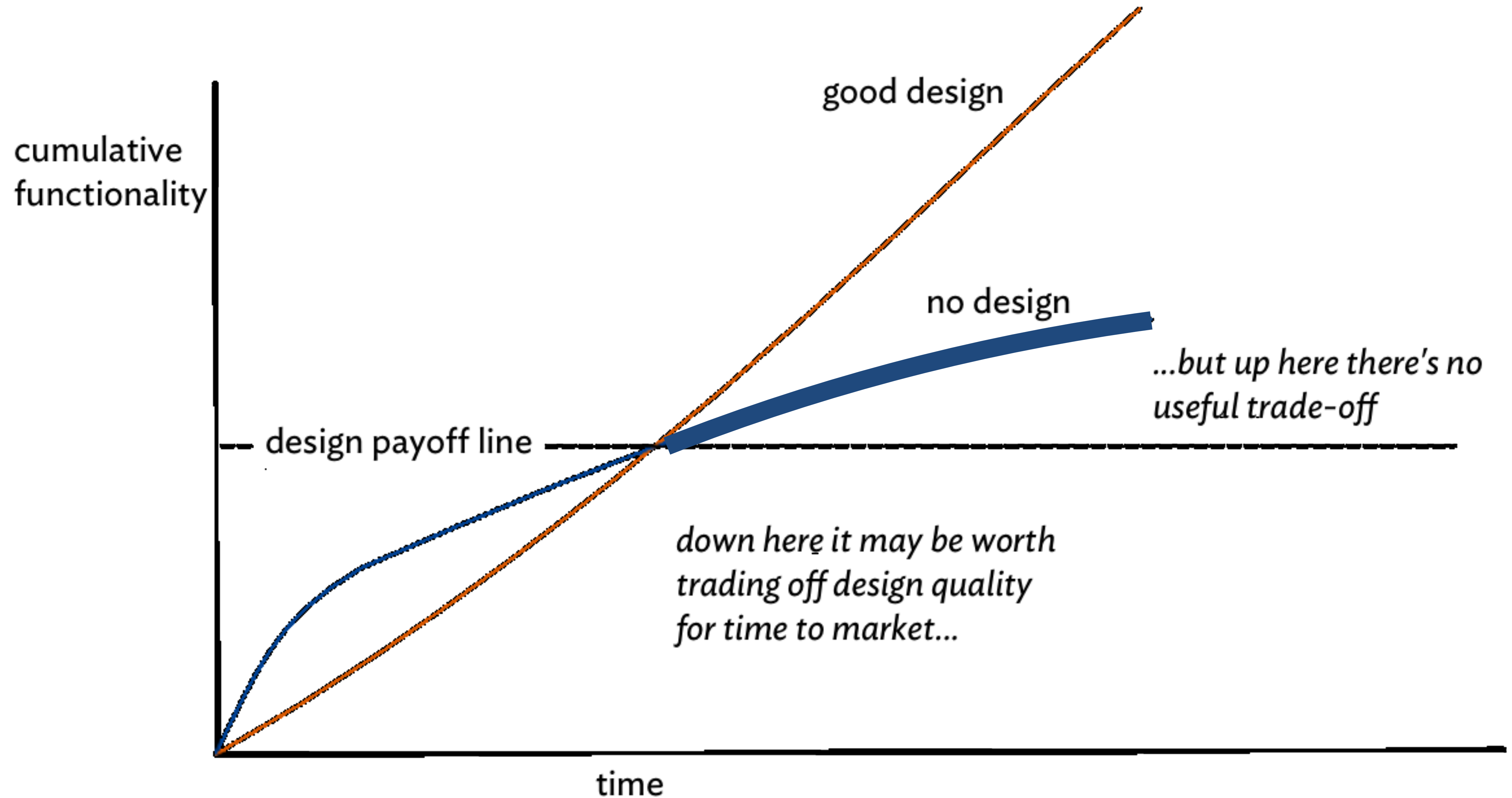






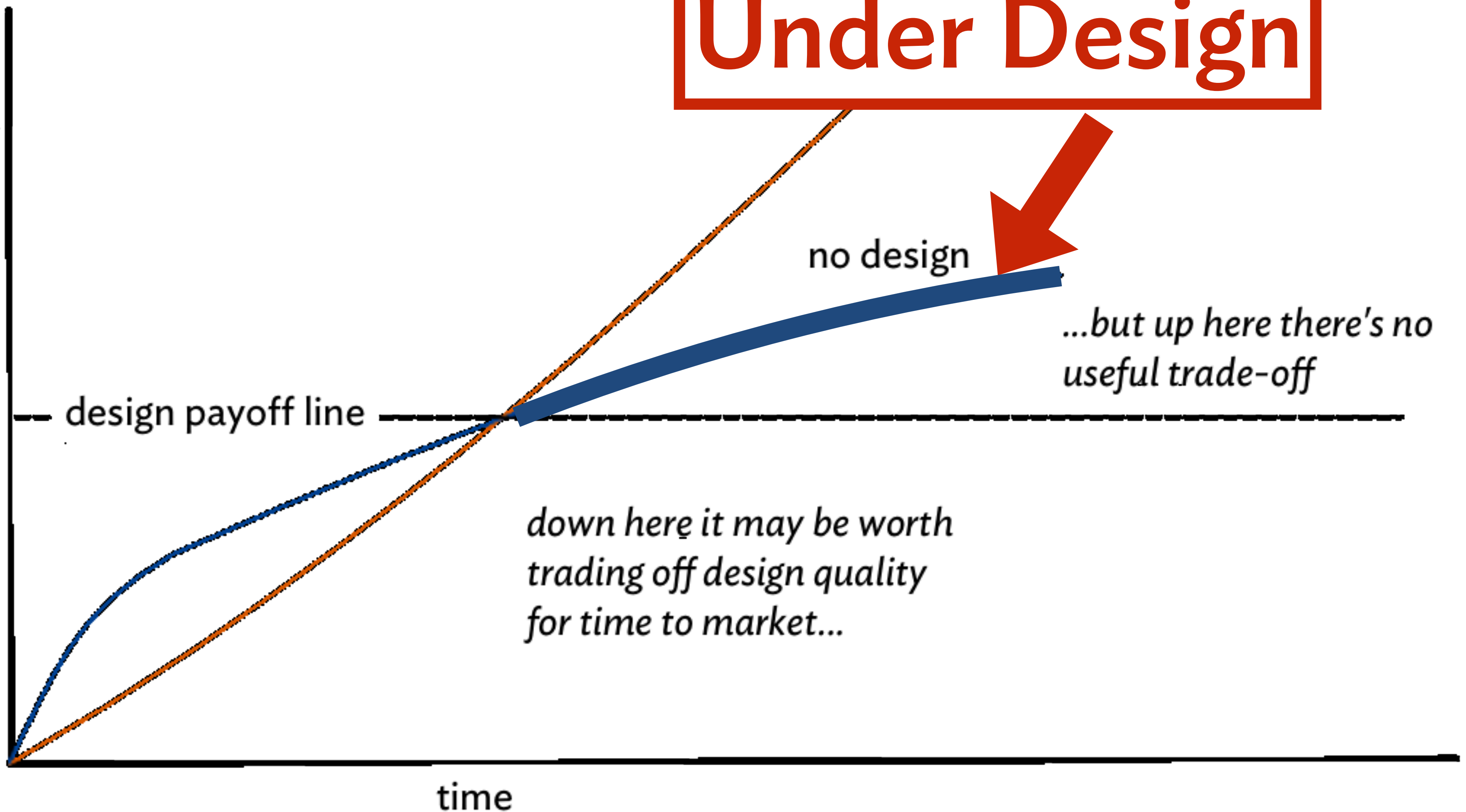


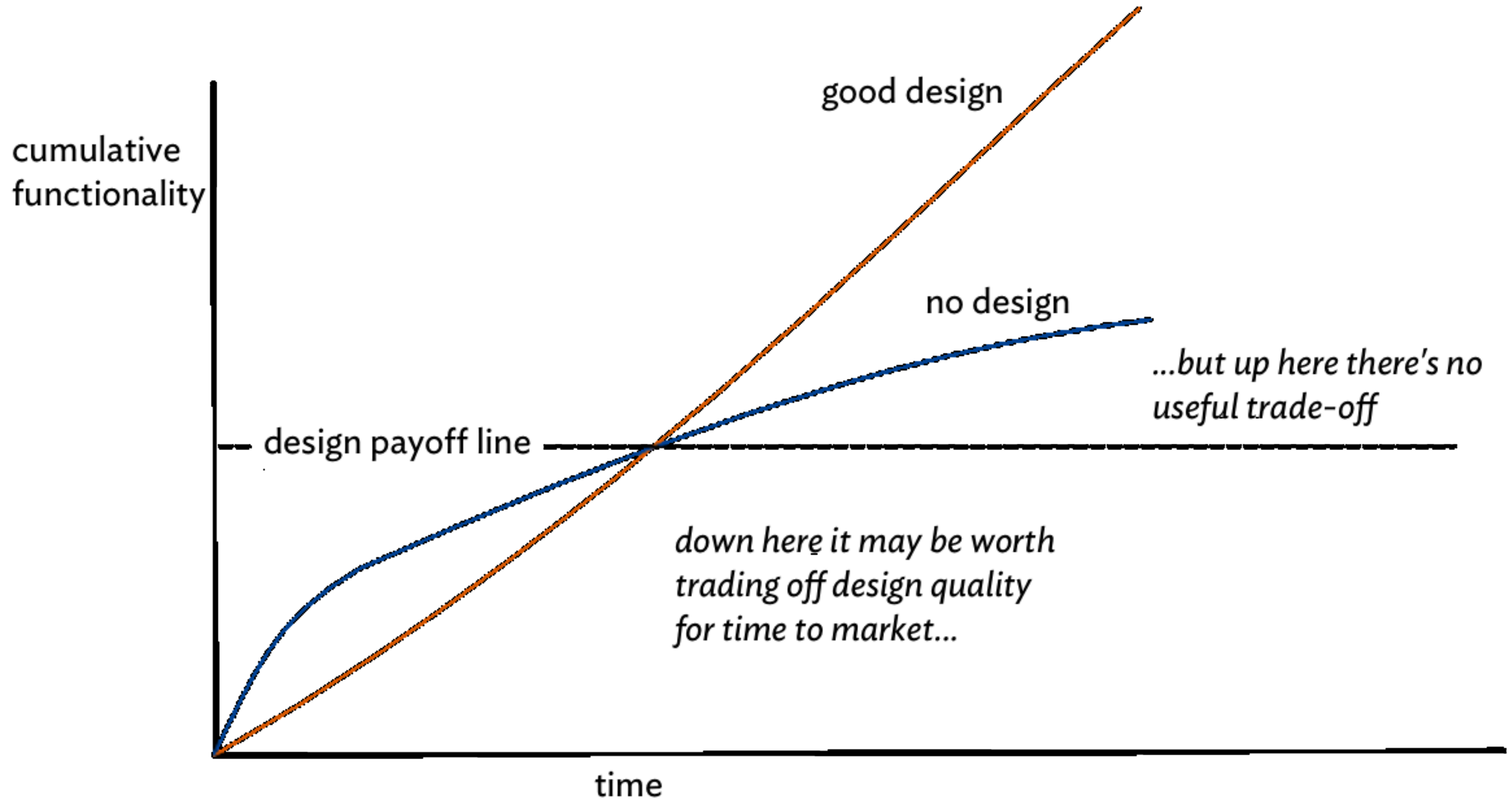


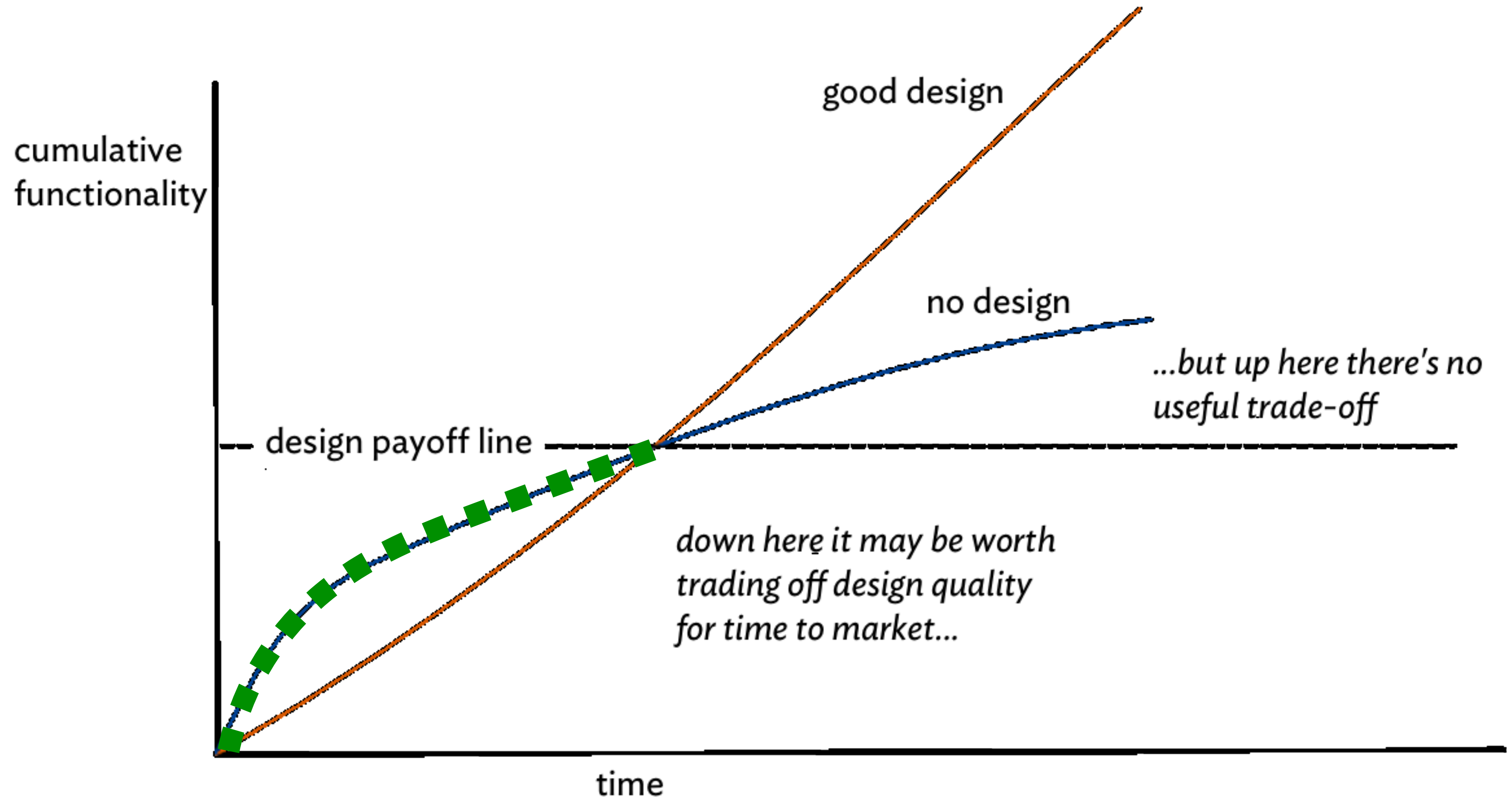


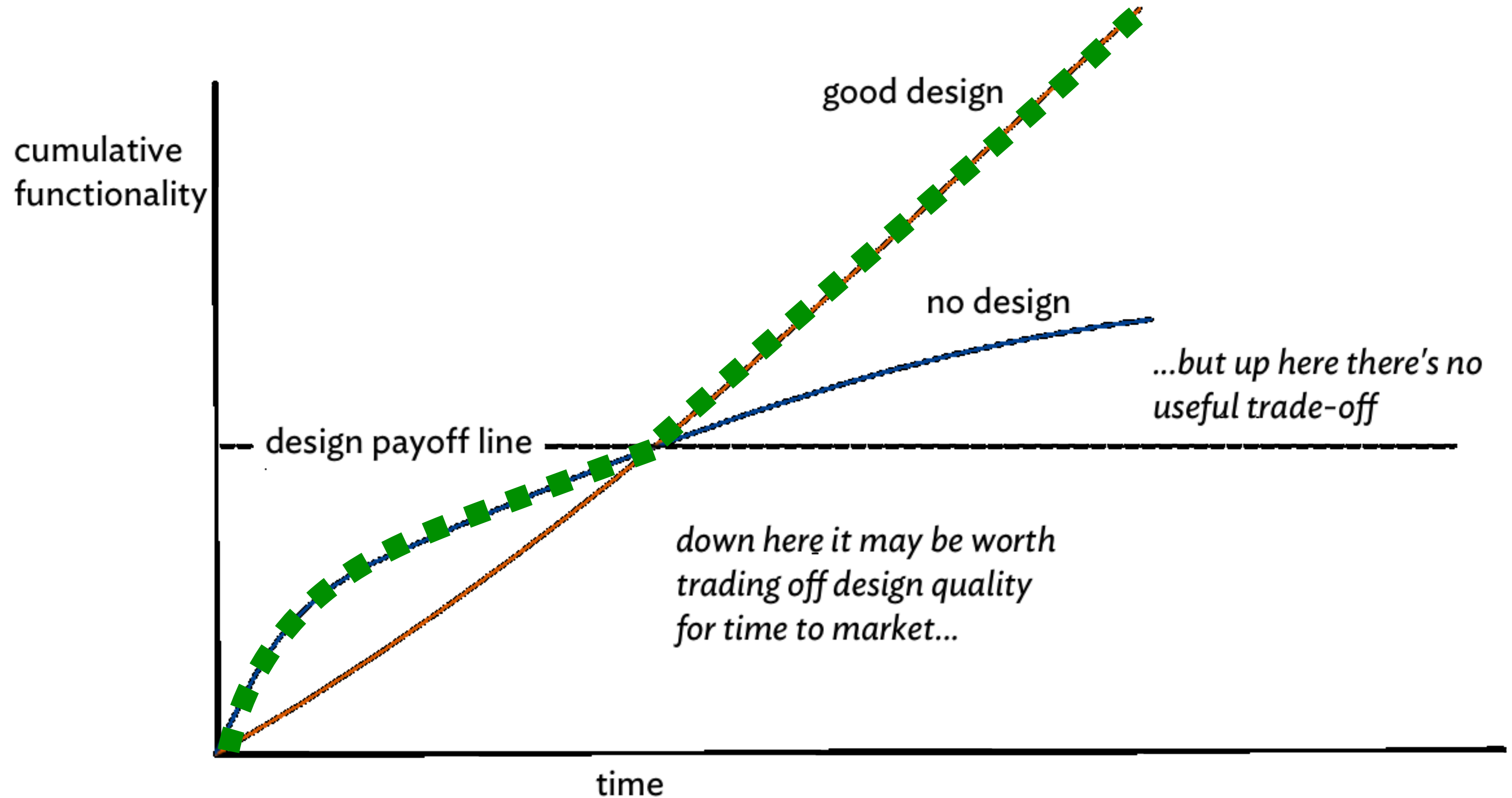
Under Design

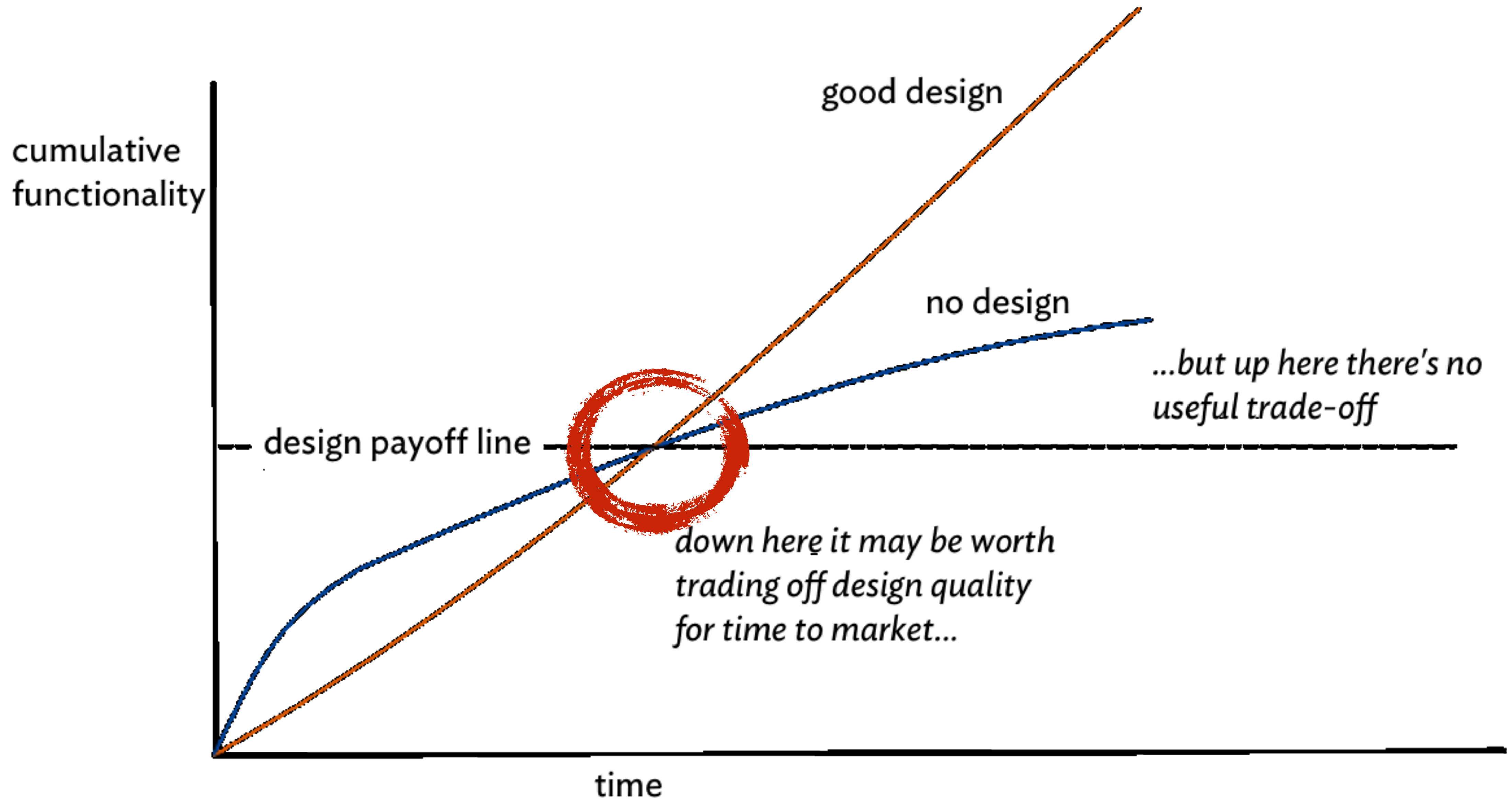
cumulative
functionality

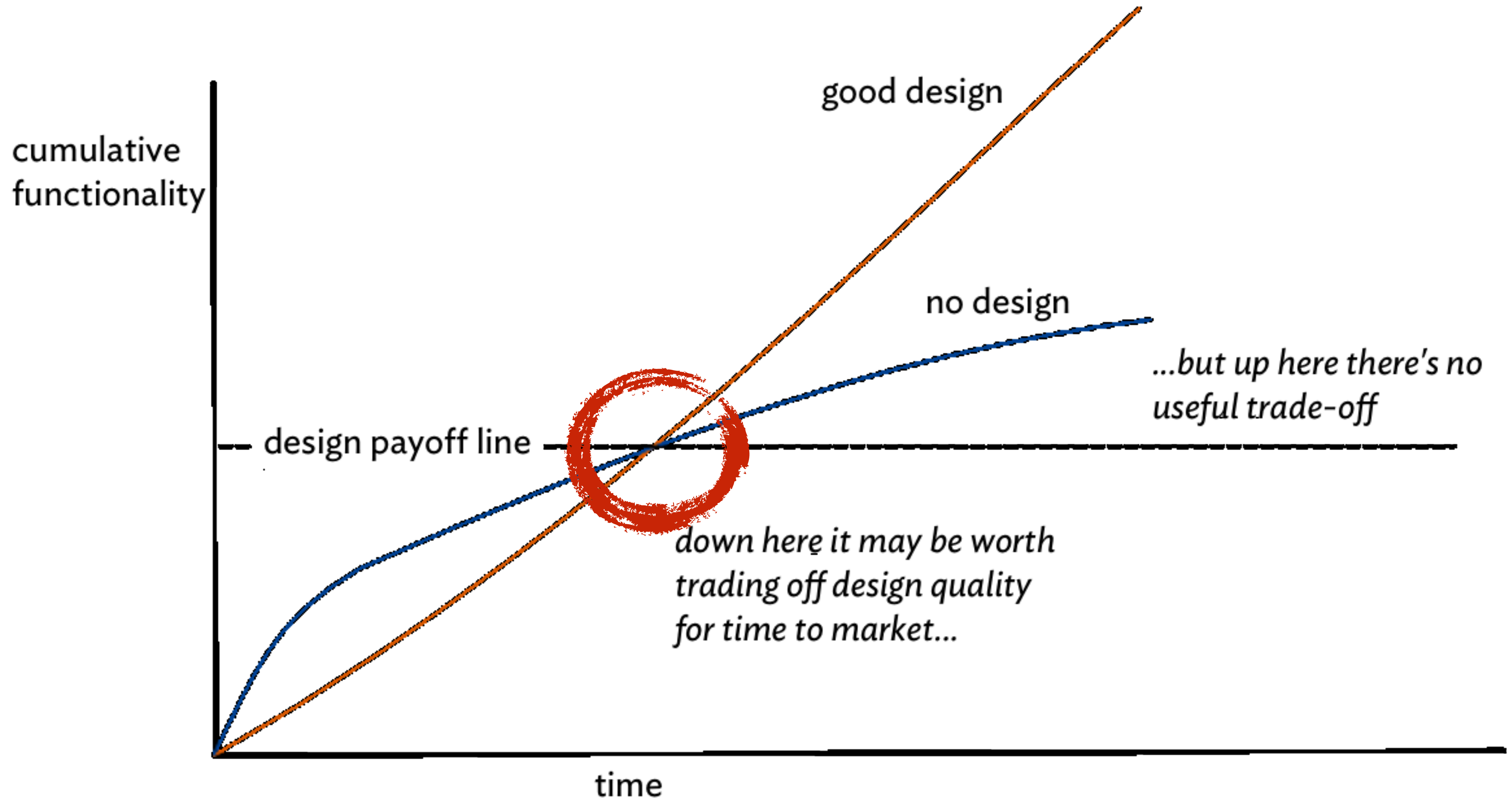


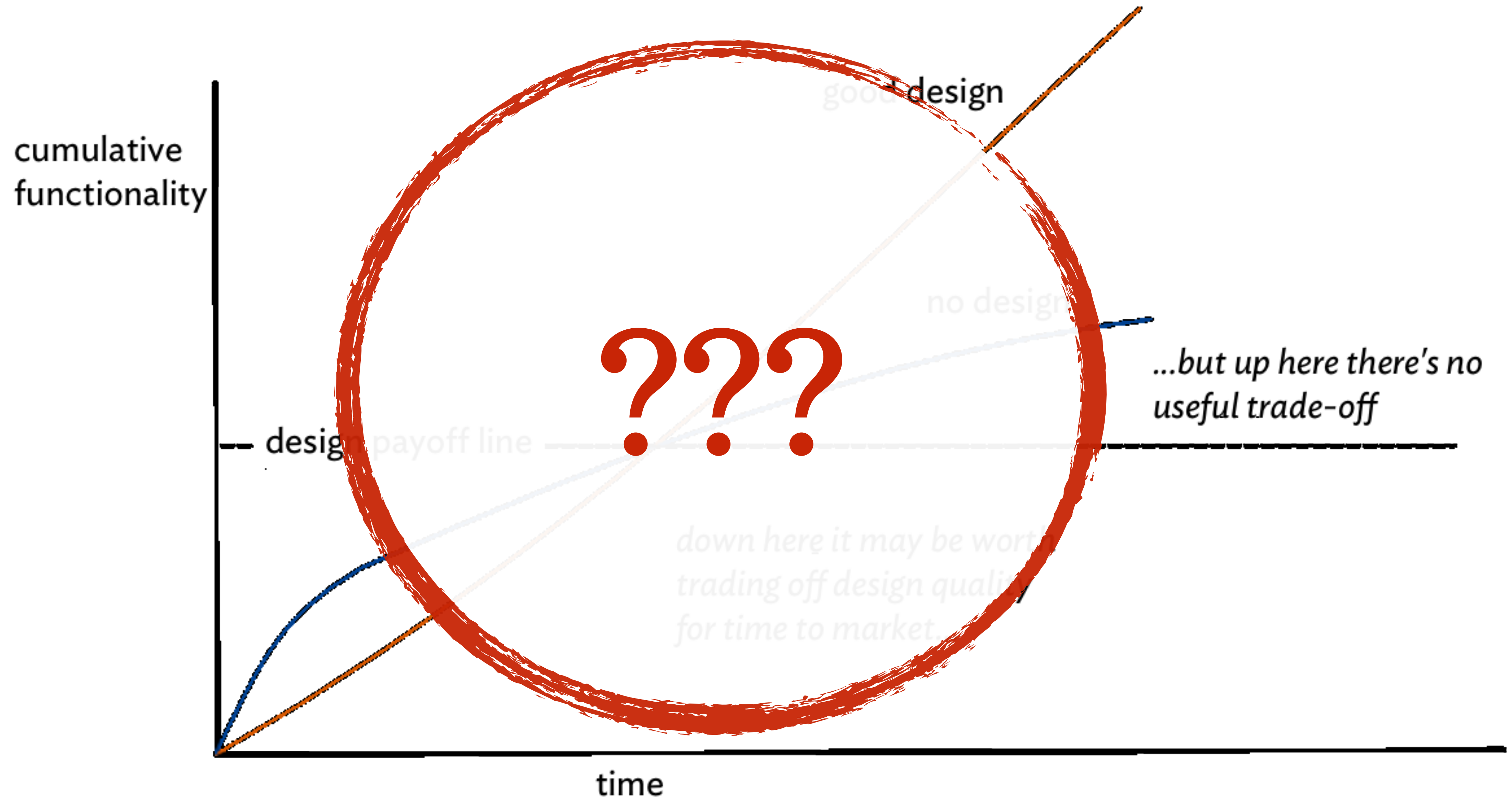


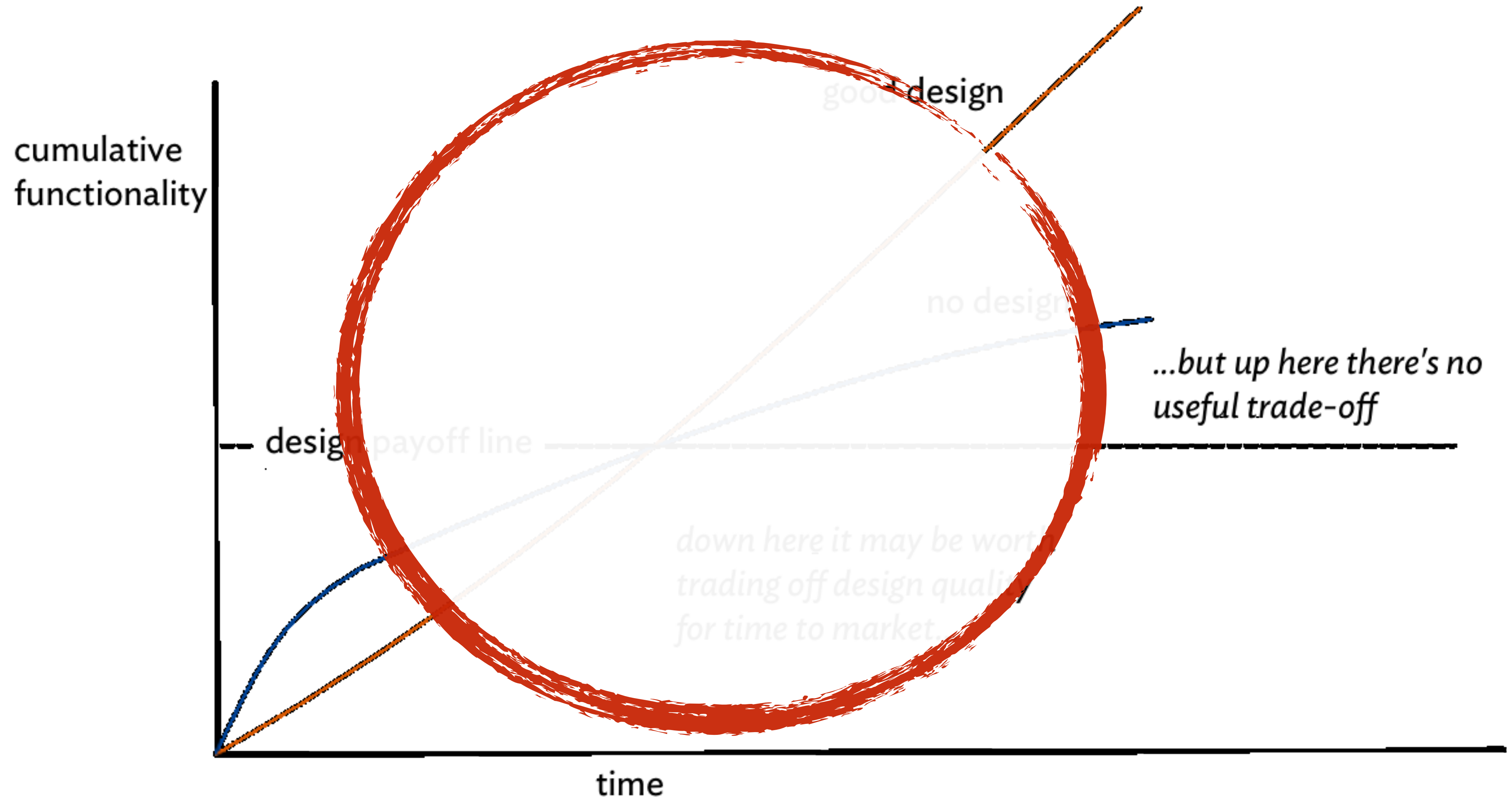


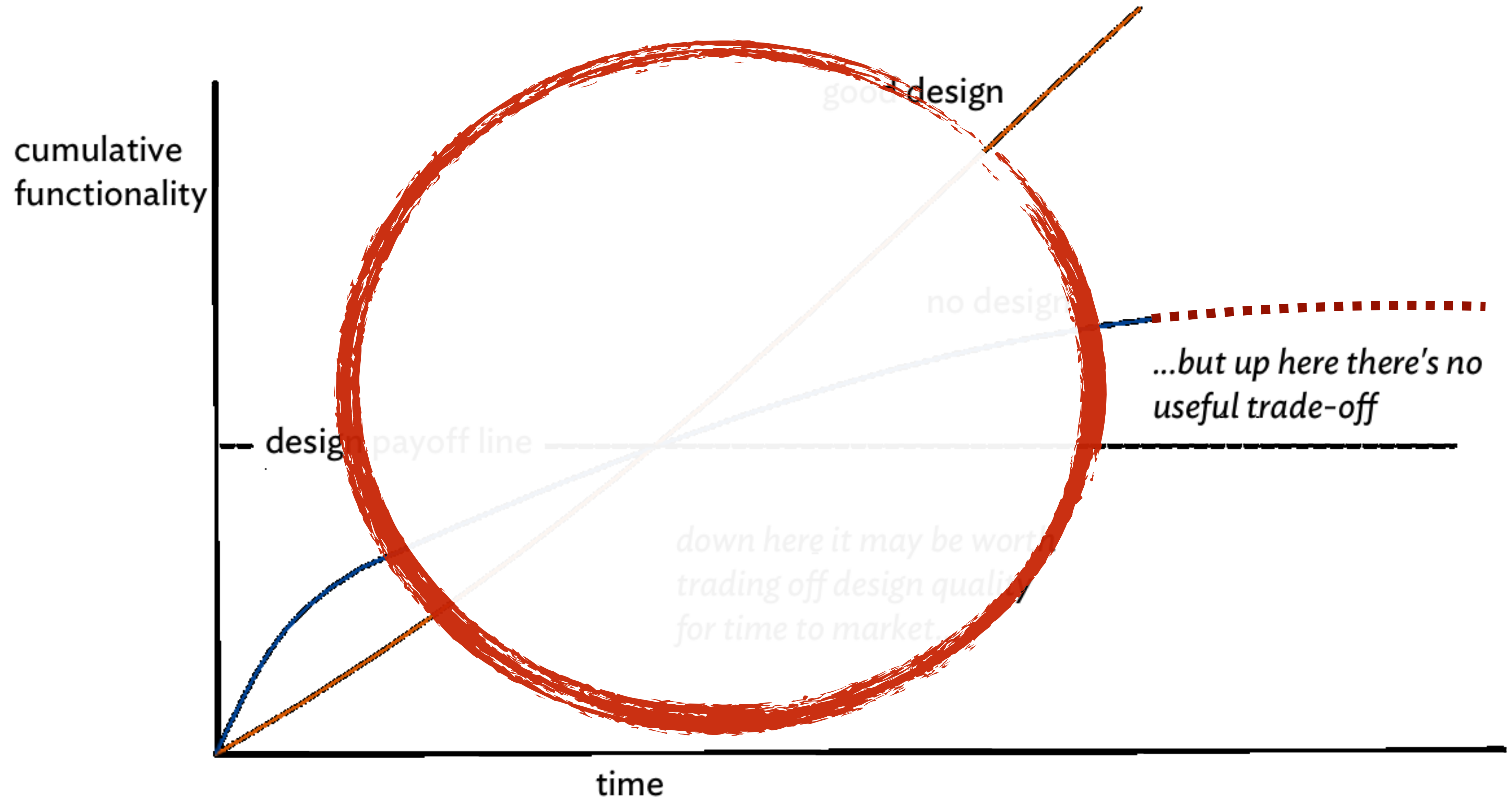


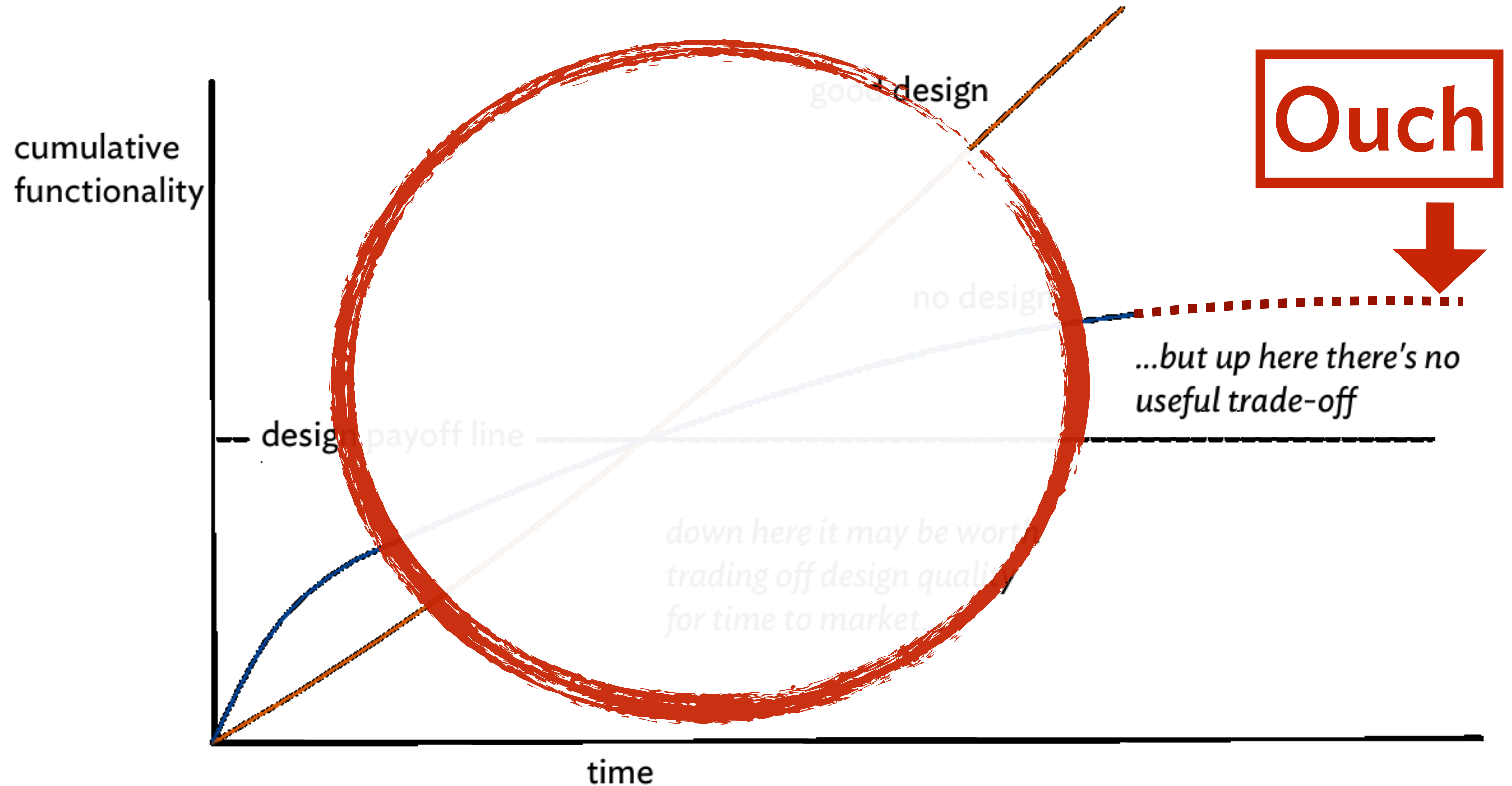












Procedures vs OO

-Me



Procedures vs OO

-Me

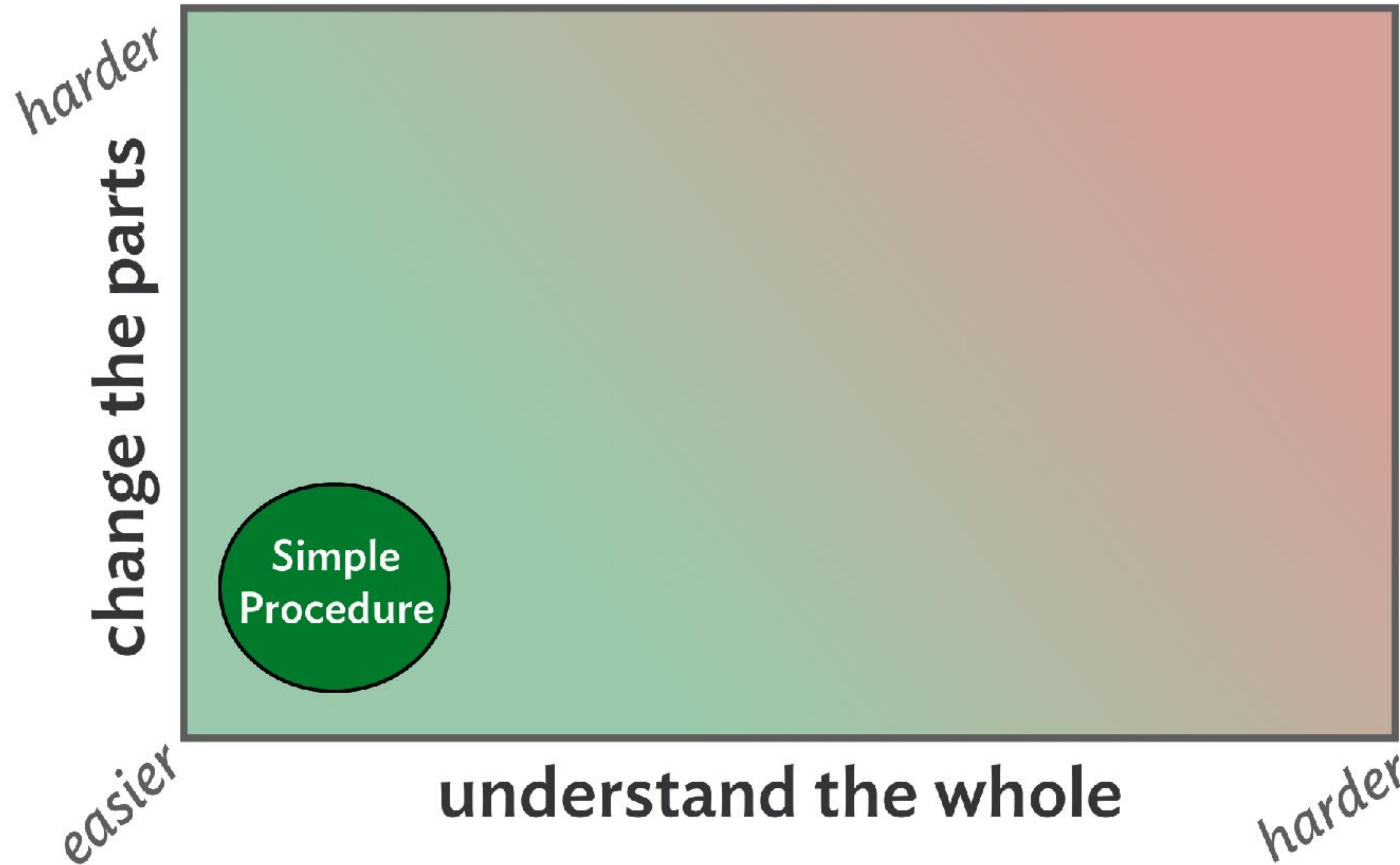


#2

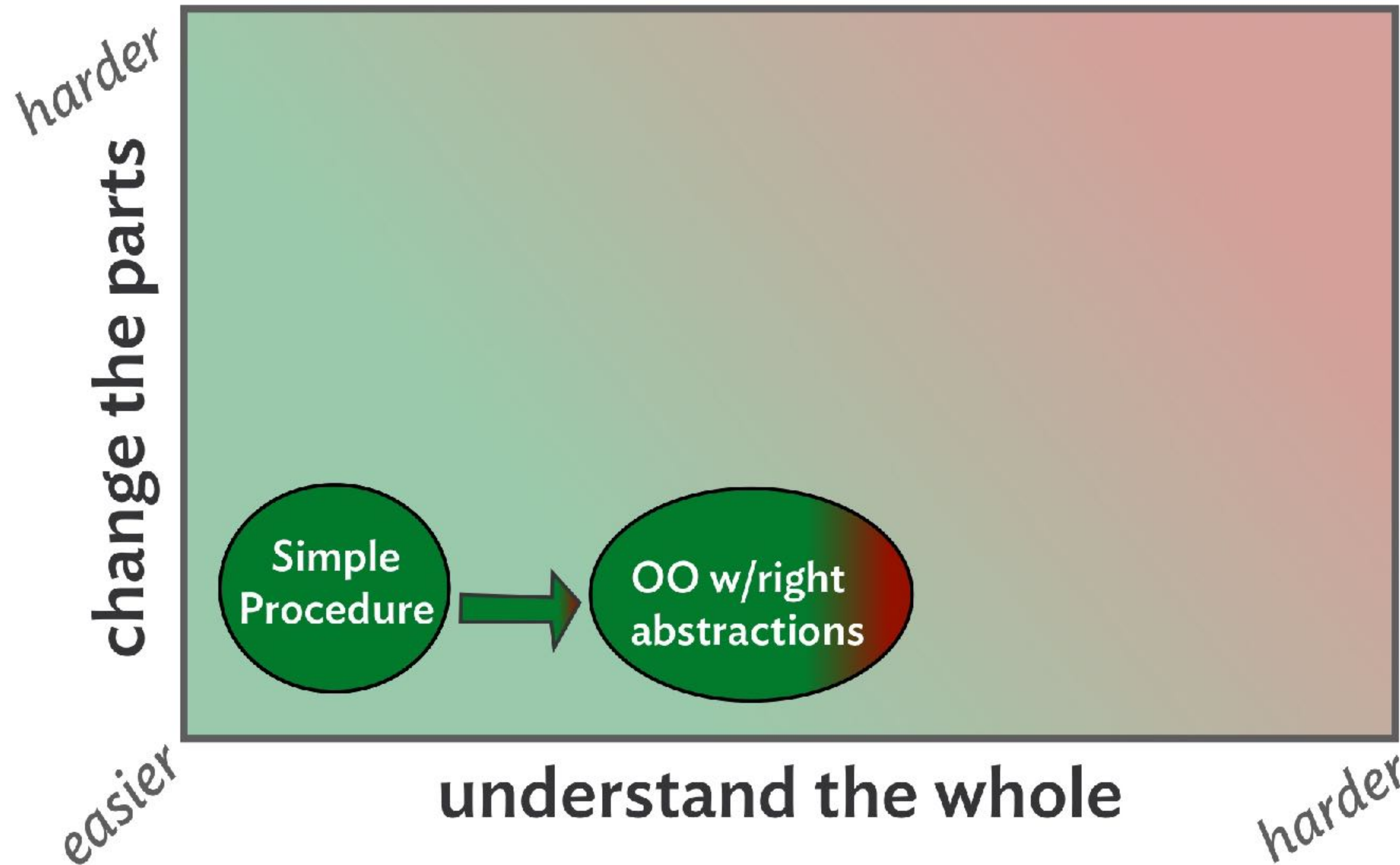
Procedural vs Object-Oriented Code



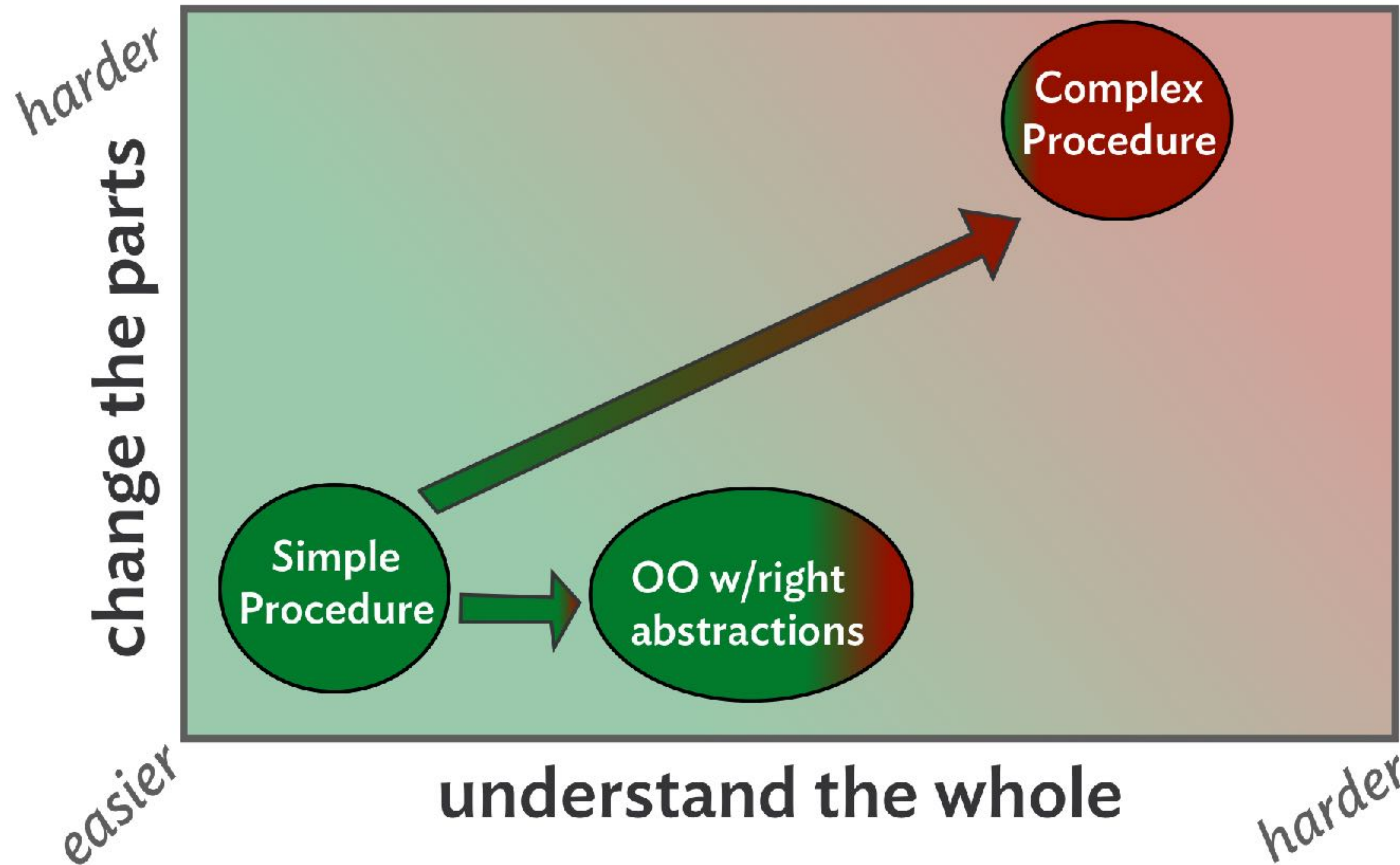
Procedural vs Object-Oriented Code



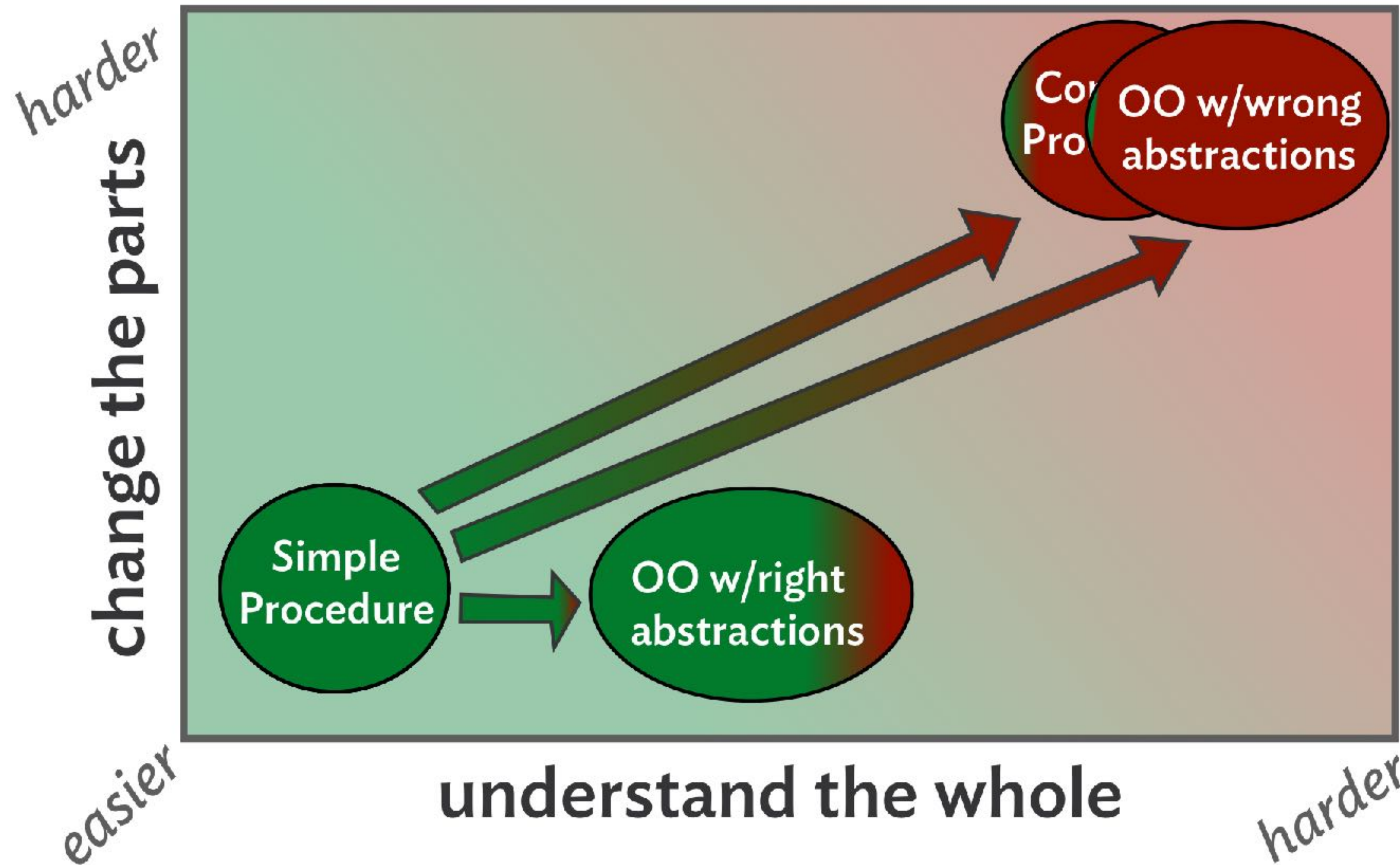
Procedural vs Object-Oriented Code



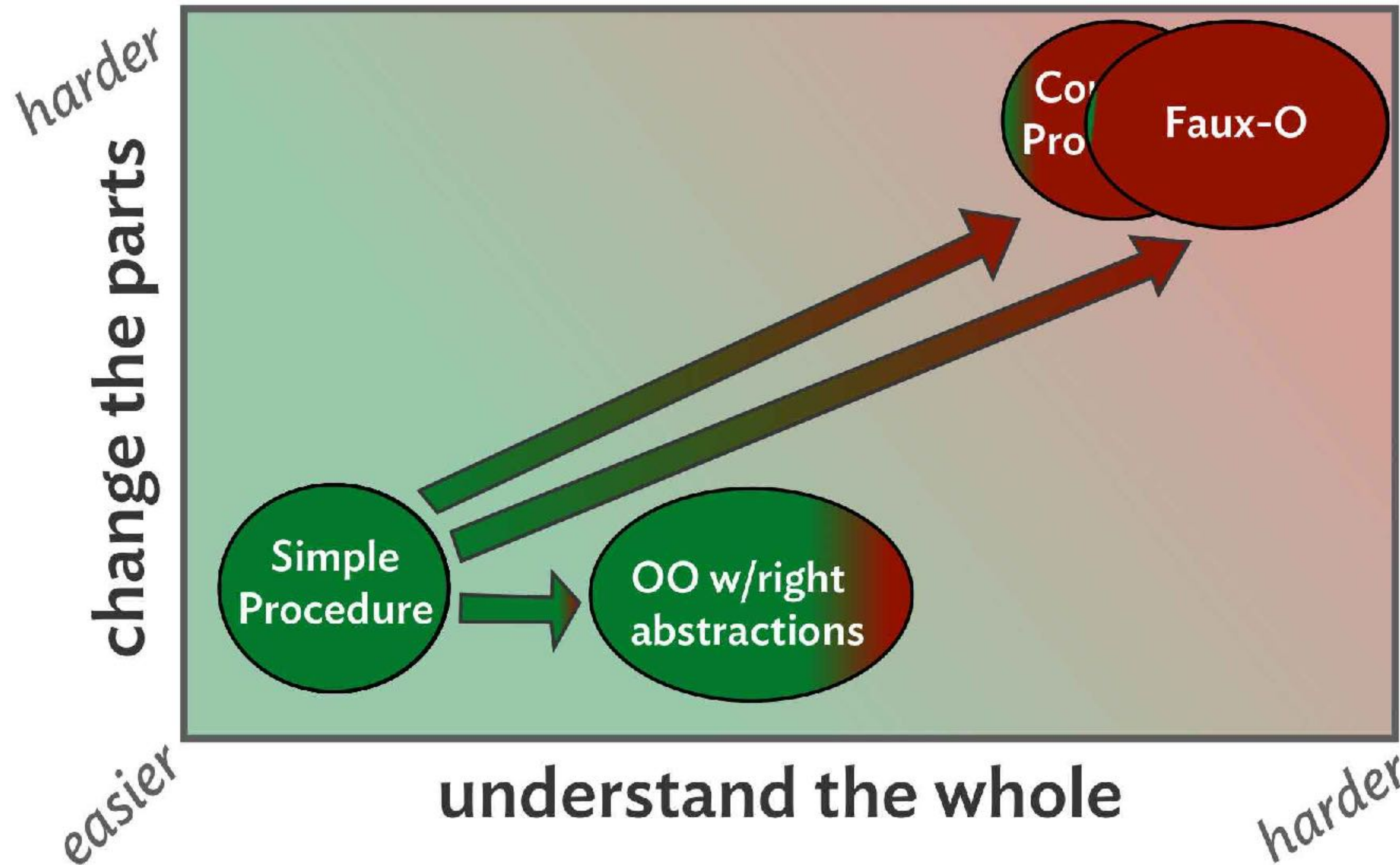
Procedural vs Object-Oriented Code



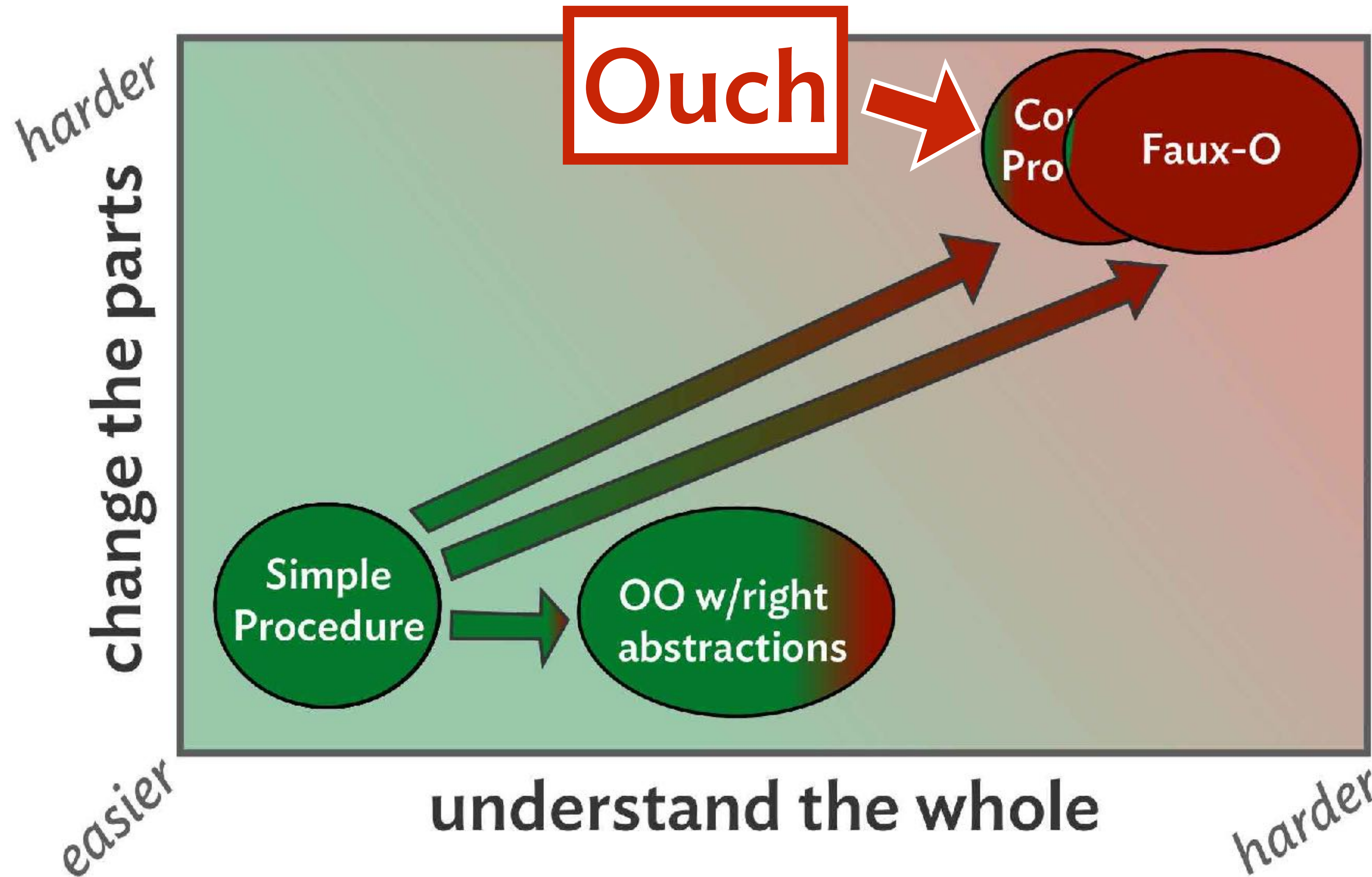
Procedural vs Object-Oriented Code



Procedural vs Object-Oriented Code



Procedural vs Object-Oriented Code



Churn vs. Complexity

– Michael Feathers



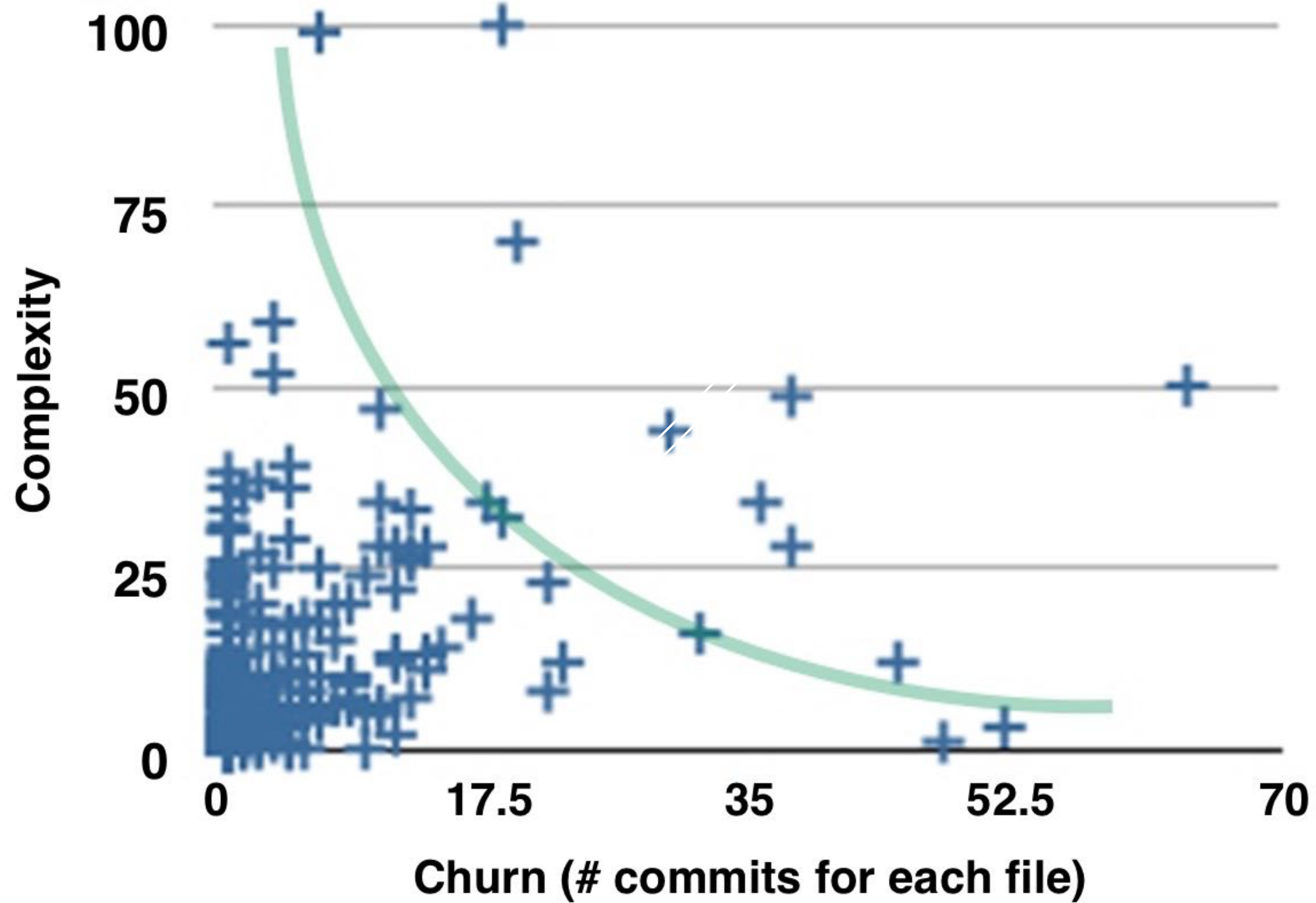
Churn vs. Complexity

– Michael Feathers

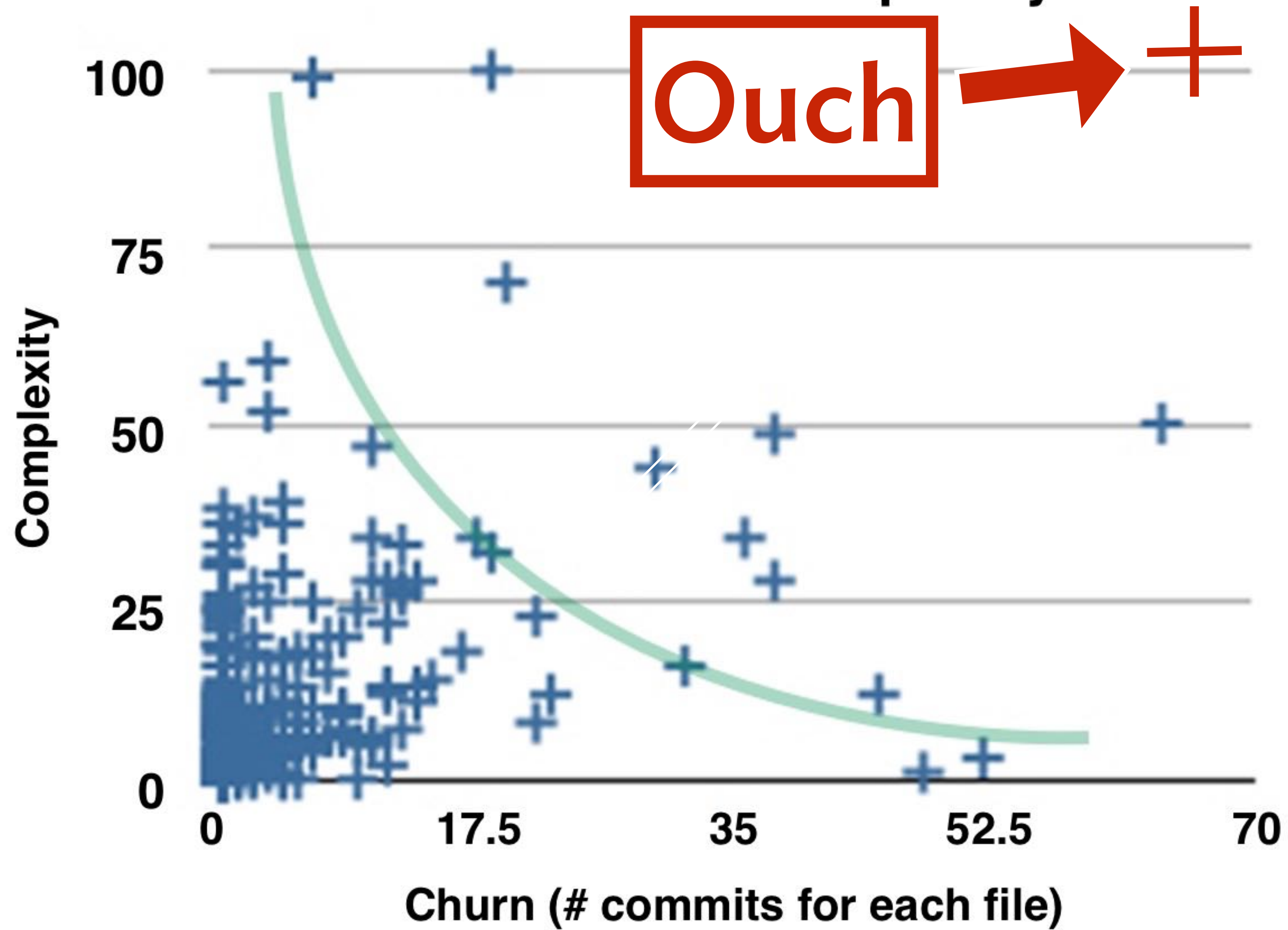


#3

File Churn vs. Complexity



File Churn vs. Complexity



What matters, suffers

-Me



What matters, suffers

-Me



#4

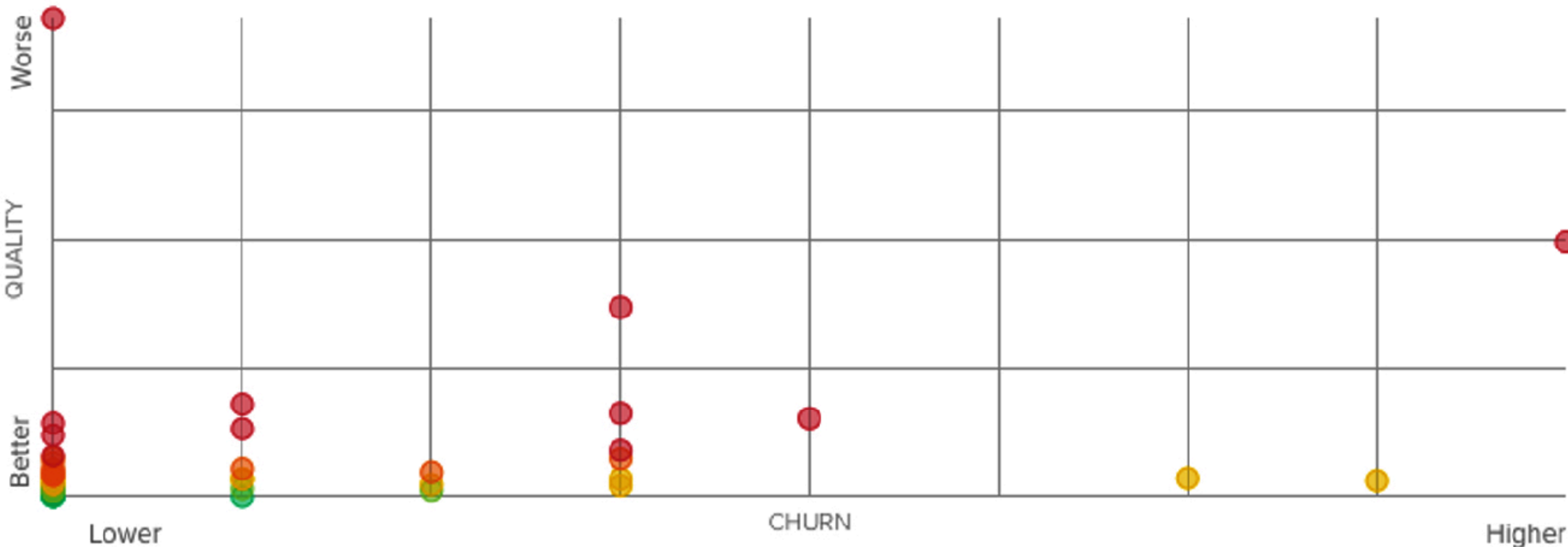
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

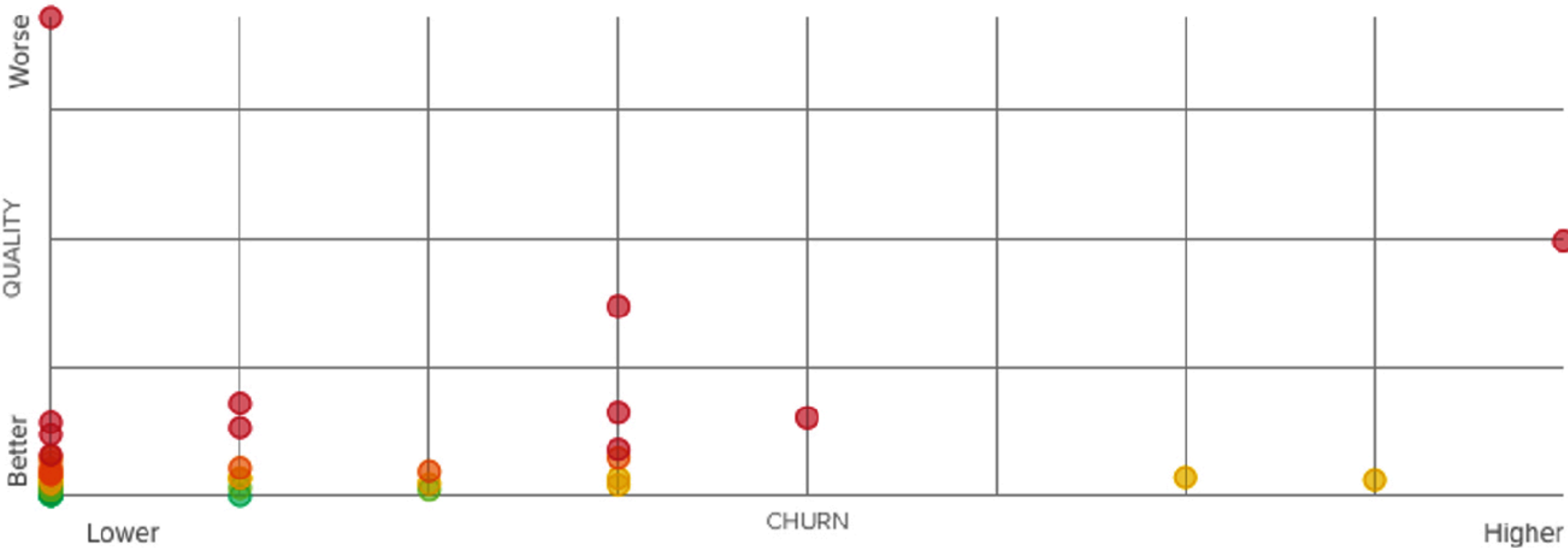
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 1.8

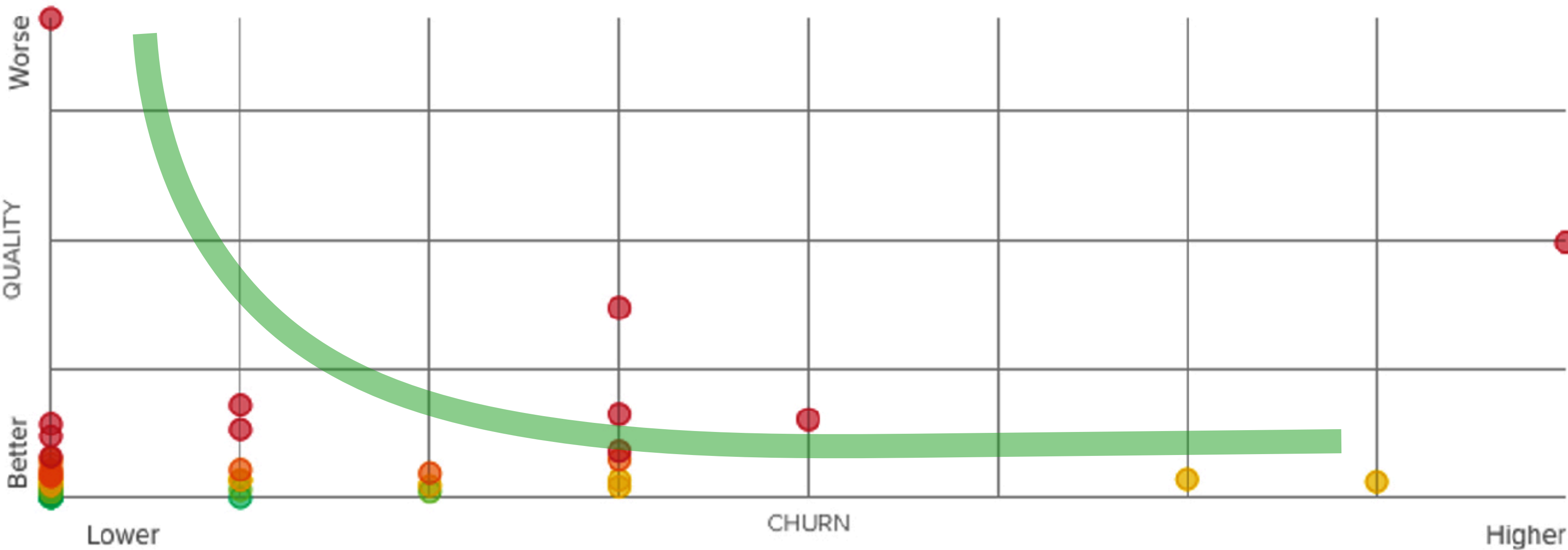
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 1.8

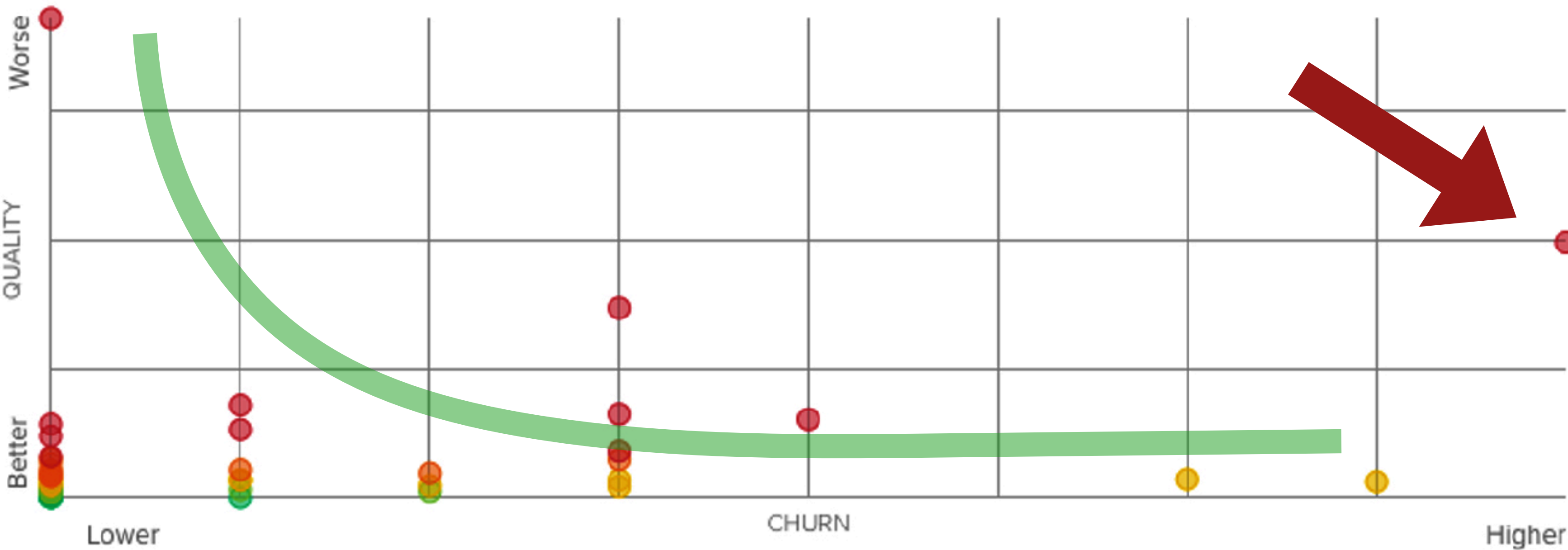
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



GPA 1.8

For performance, we've limited this chart to 50 data points per rating.

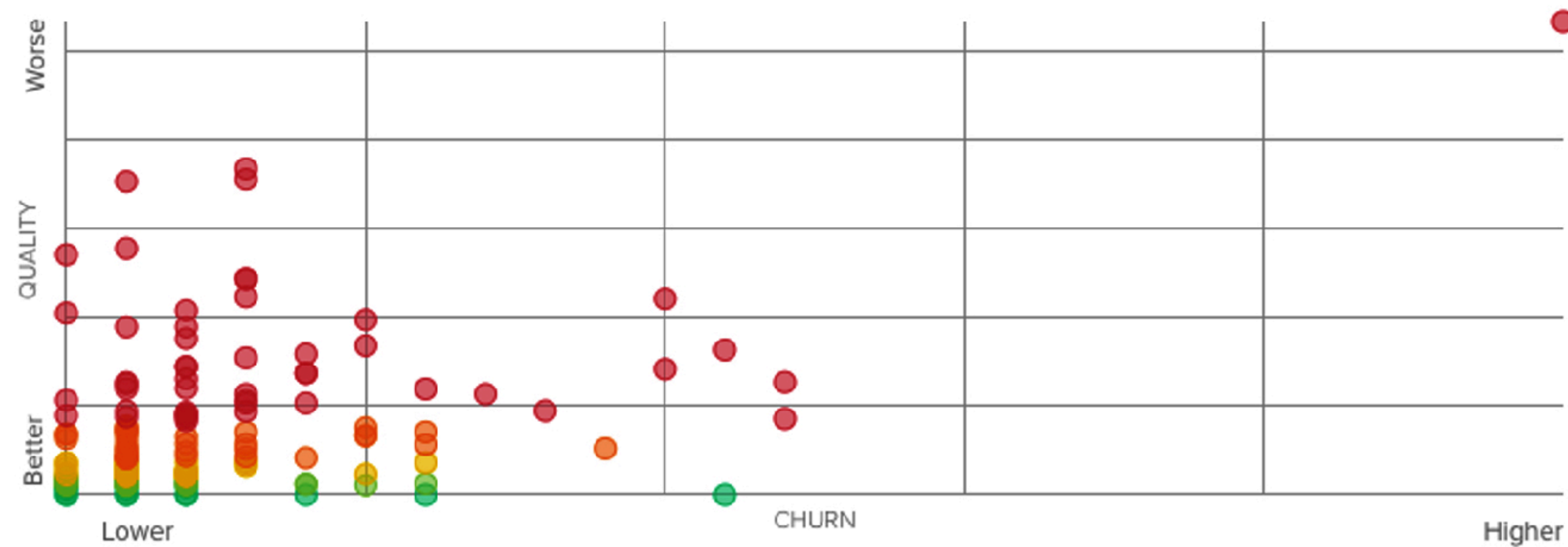
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

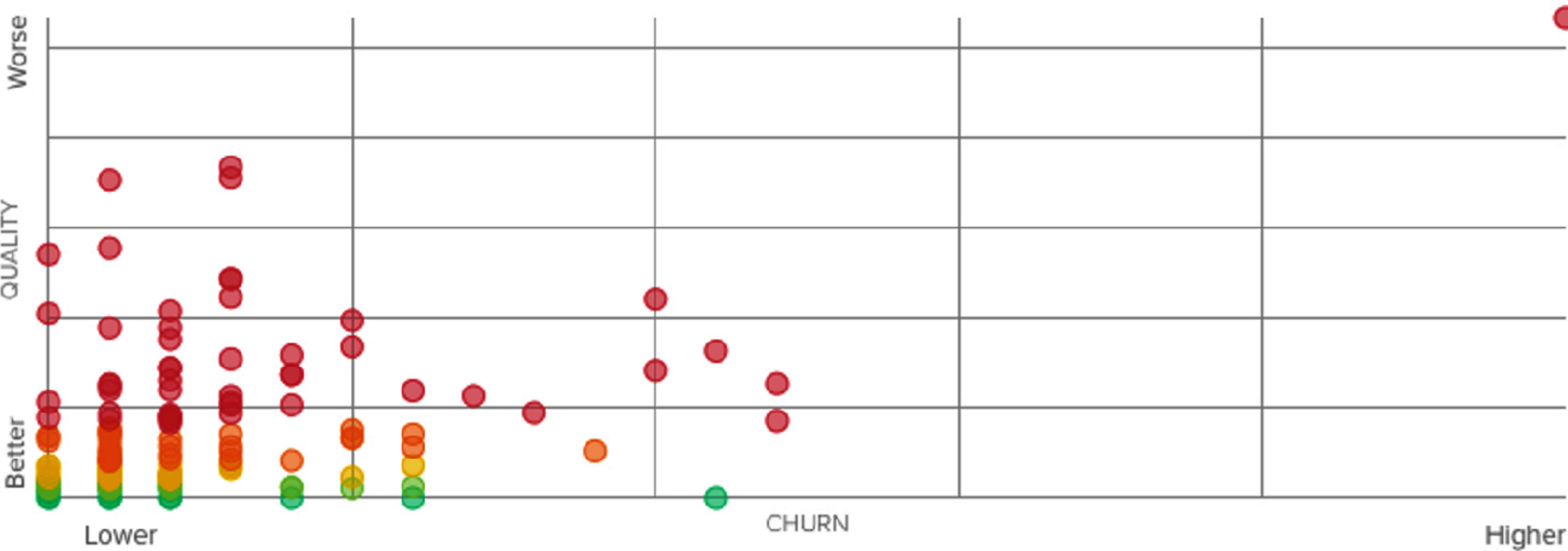
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 2.58

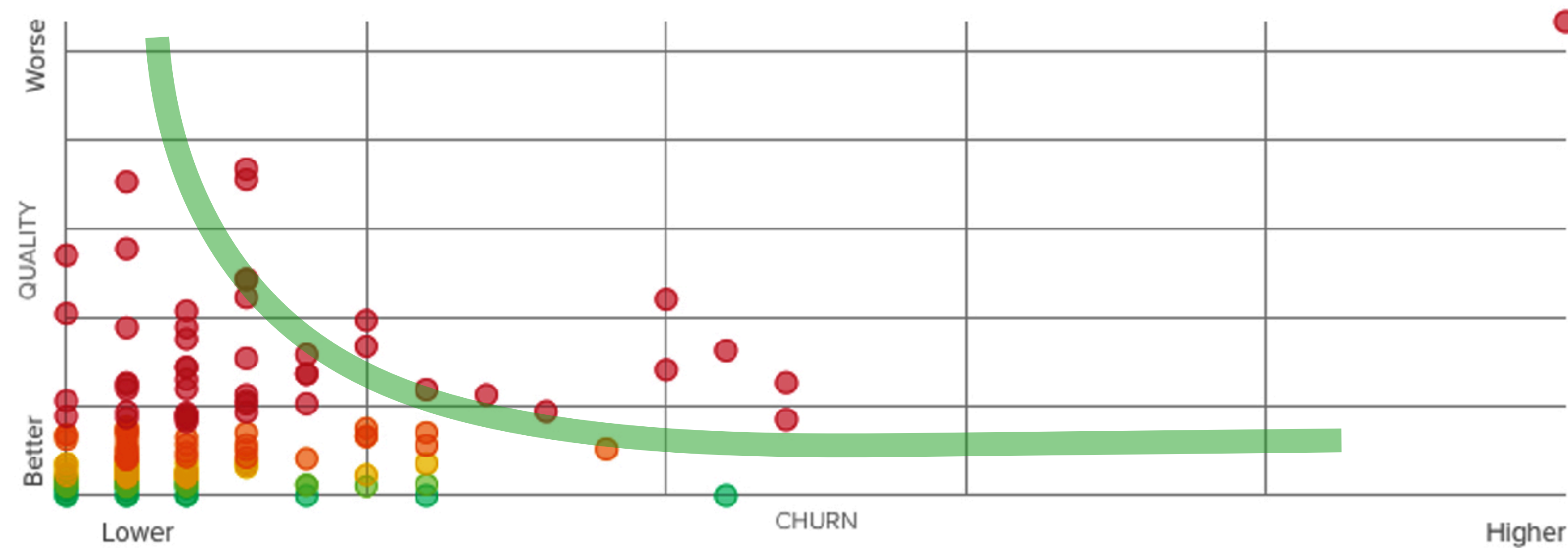
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 2.58

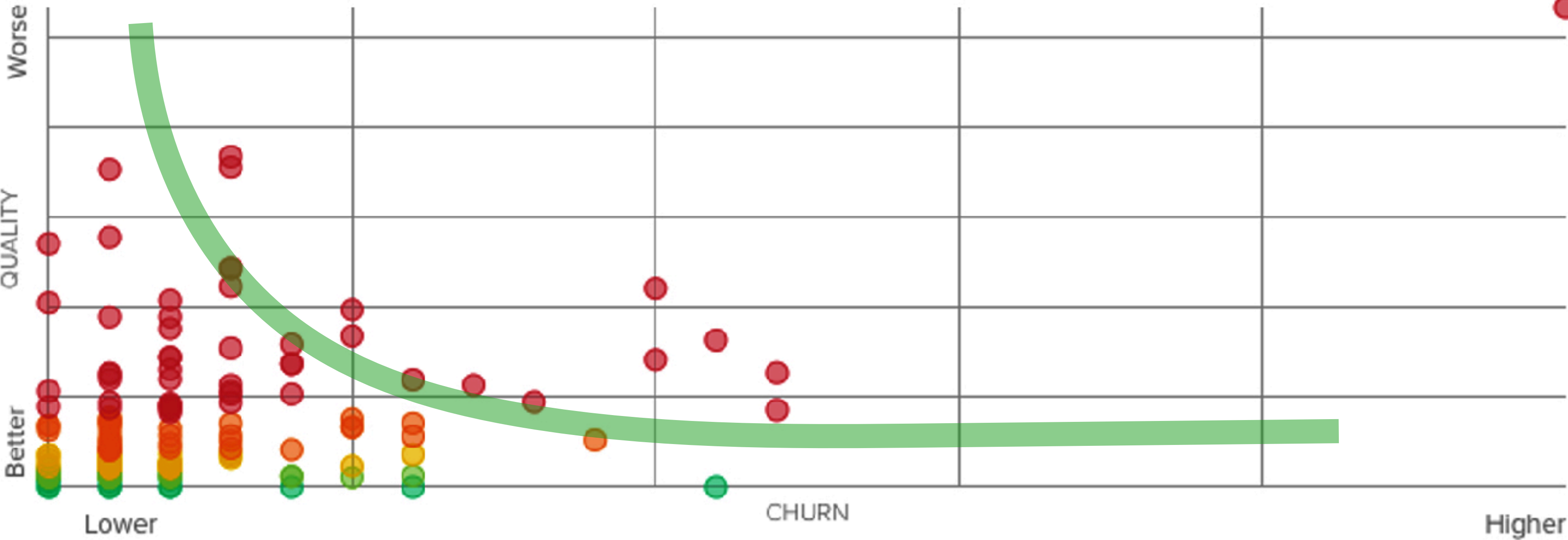
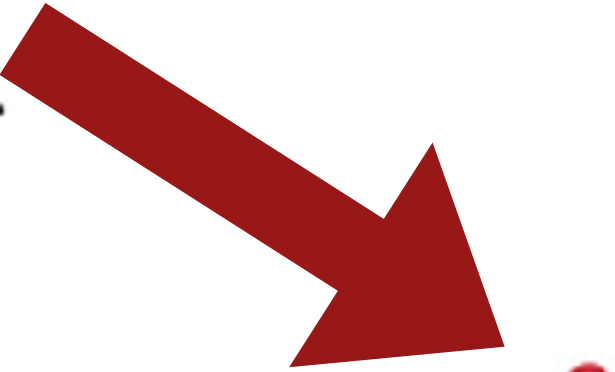
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 2.58

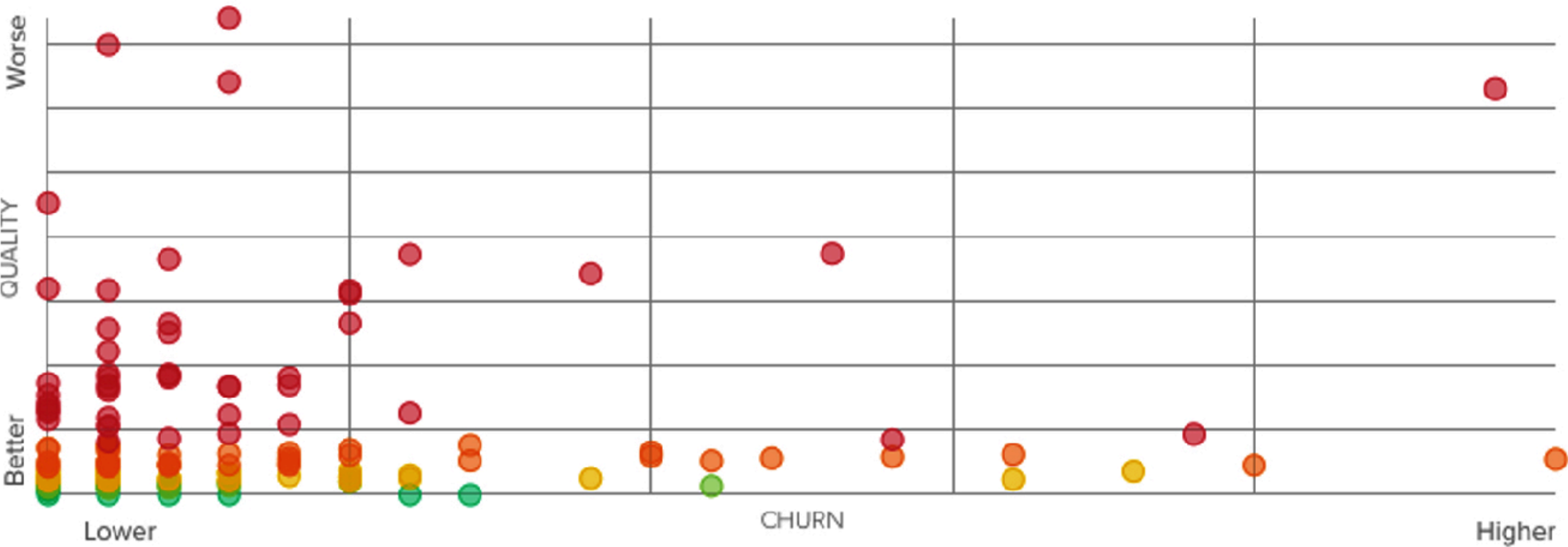
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

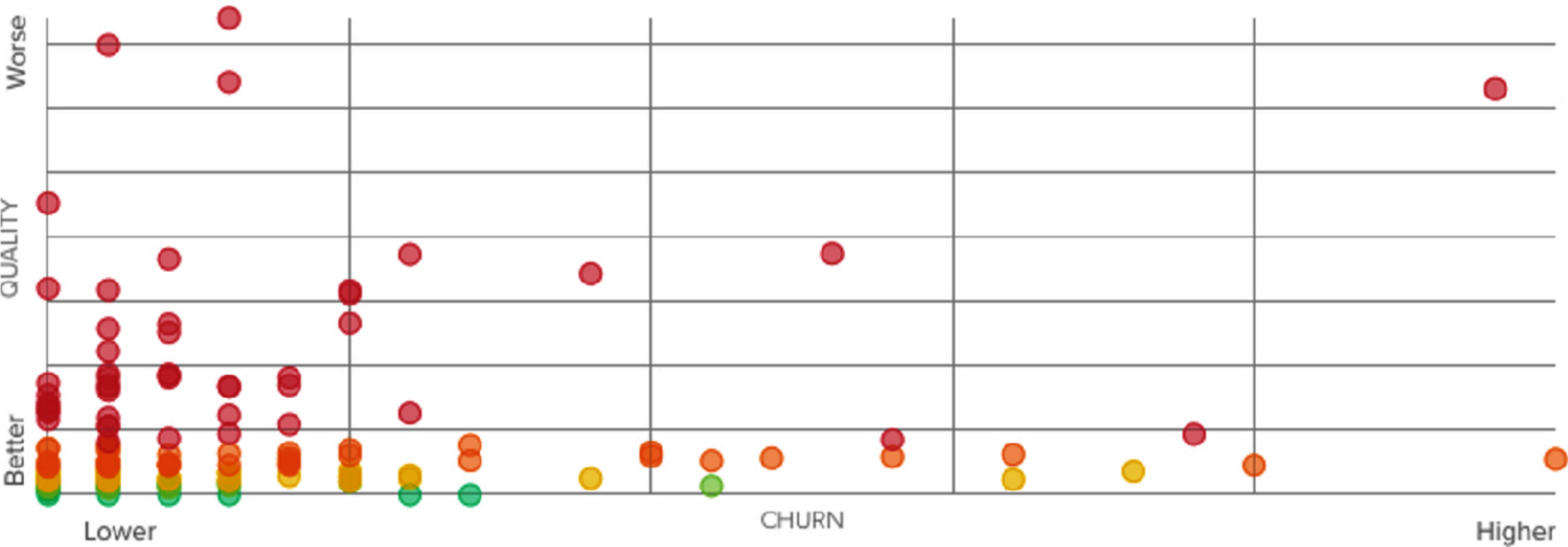
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 2.97

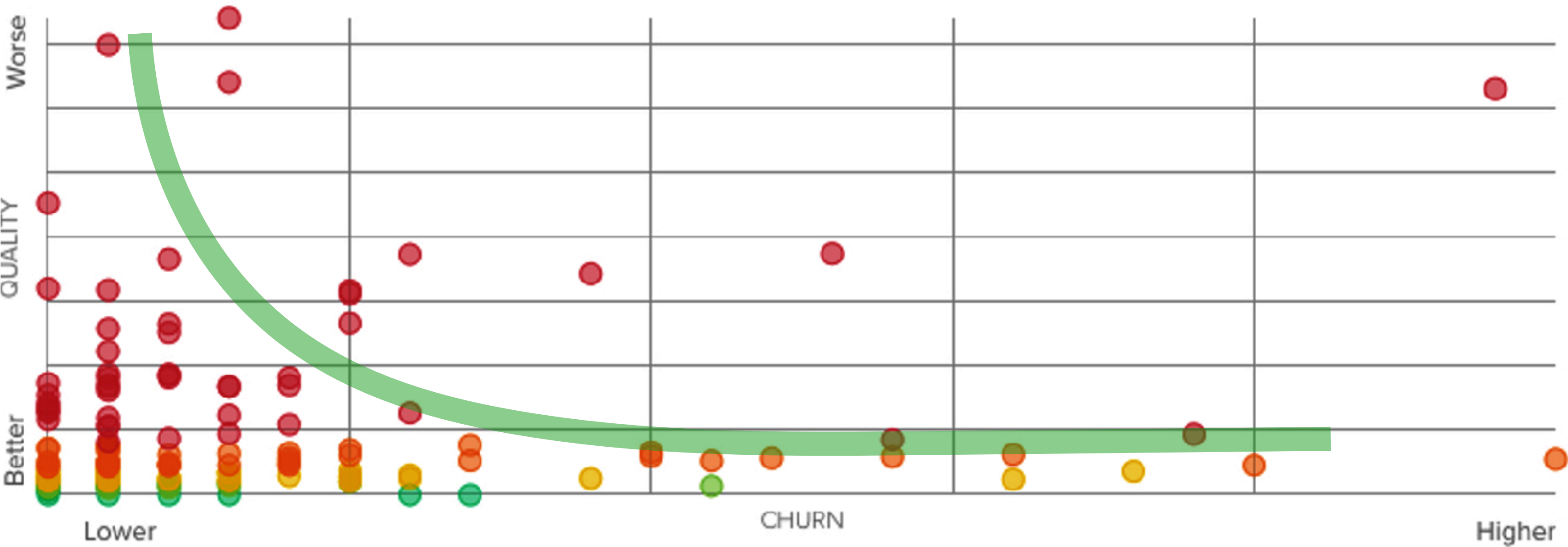
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 2.97

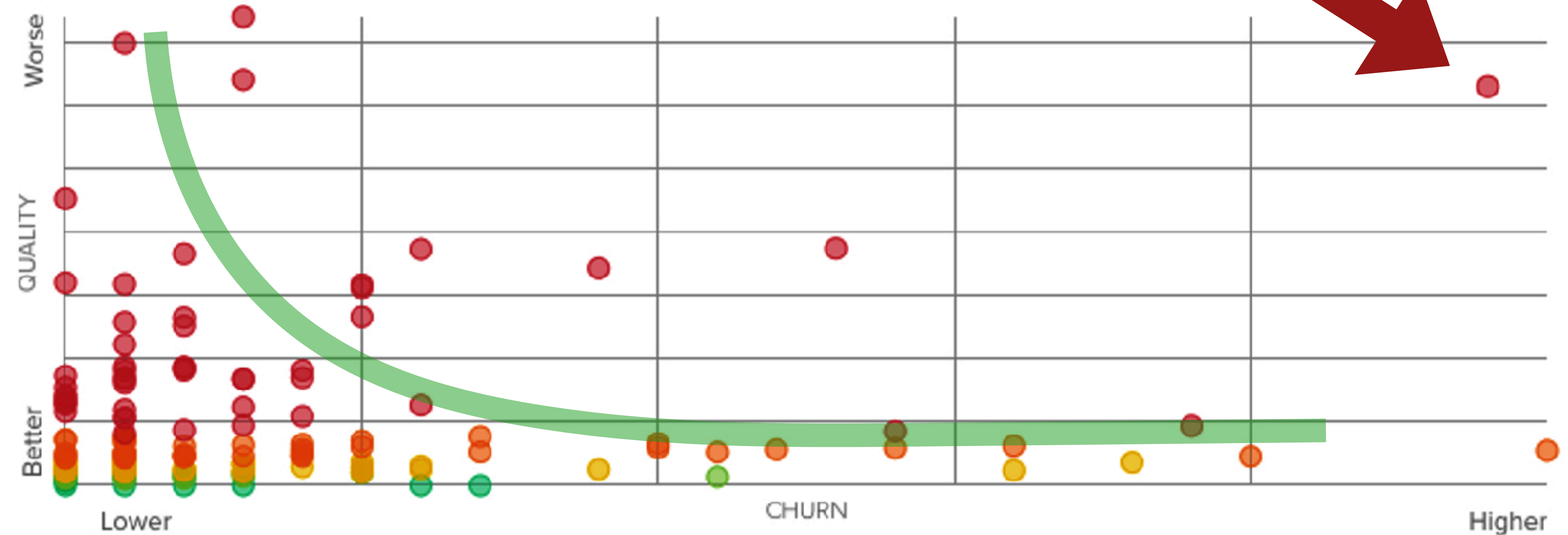
CHARTS

GPA

Churn vs. quality

Churn vs. quality

Quality issues cause bigger problems in files that are changed (churn) frequently.



For performance, we've limited this chart to 50 data points per rating.

GPA 2.97

CHARTS

GPA

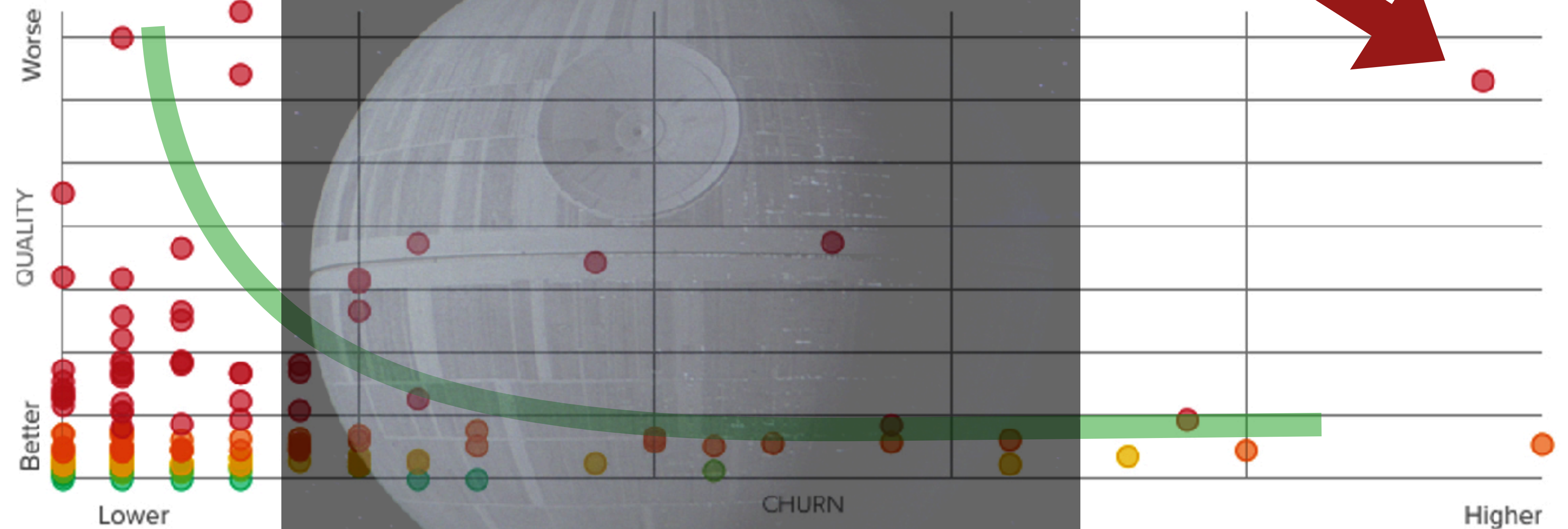
Churn vs. quality

Churn
Quality is

Death Star

Anti-Pattern

GPA 2.97



For performance, we've limited this chart to 50 data points per rating.



#1 Design Stamina Hypothesis



#1 Design Stamina Hypothesis

#2 Procedures vs OO



- #1 Design Stamina Hypothesis
- #2 Procedures vs OO
- #3 Churn vs Complexity



- #1 Design Stamina Hypothesis
- #2 Procedures vs OO
- #3 Churn vs Complexity
- #4 What matters, suffers



Interlude

Interlude

*Wherein things go
badly wrong*

```
File.read("/path/to/filename").split("\n")
```

Change #1

Change #1

*Read from
Git Tag*


```
class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir="#{repository}"]
  end
end
end
```

```
class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir="#{repository}"]
  end
end
end
```

```
class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir="#{repository}"]
  end
end
end
```

```
class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir="#{repository}"]
  end
end
end
```



```
if git_cmd
  git_cmd.repository = repository
  git_cmd.tagname    = tag
  git_cmd.filename   = filename
  git_cmd.show.split("\n")
else
  File.read(filename).split("\n")
end
```

```
if git_cmd
  git_cmd.repository = repository
  git_cmd.tagname    = tag
  git_cmd.filename   = filename
  git_cmd.show.split("\n")
else
  File.read(filename).split("\n")
end
```

```
if git_cmd
  git_cmd.repository = repository
  git_cmd.tagname    = tag
  git_cmd.filename   = filename
  git_cmd.show.split("\n")
else
  File.read(filename).split("\n")
end
```

```
def lines
  if git_cmd
    git_cmd.repository = repository
    git_cmd.tagname     = tag
    git_cmd.filename    = filename
    git_cmd.show.split("\n")
  else
    File.read(filename).split("\n")
  end
end
```



```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end
end
```

class Listing

```
attr_reader :filename, :repository, :tag, :git_cmd
```

```
def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
```

```
  @filename = filename
```

```
  @repository = repository
```

```
  @tag = tag
```

```
  @git_cmd = git_cmd
```

```
end
```

```
def lines
```

```
  if git_cmd
```

```
    git_cmd.repository = repository
```

```
    git_cmd.tagname = tag
```

```
    git_cmd.filename = filename
```

```
    git_cmd.show.split("\n")
```

```
  else
```

```
    File.read(filename).split("\n")
```

```
  end
```

```
end
```

```
end
```

class Listing

```
attr_reader :filename, :repository, :tag, :git_cmd
```

```
def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
```

```
  @filename = filename
```

```
  @repository = repository
```

```
  @tag = tag
```

```
  @git_cmd = git_cmd
```

```
end
```

```
def lines
```

```
  if git_cmd
```

```
    git_cmd.repository = repository
```

```
    git_cmd.tagname = tag
```

```
    git_cmd.filename = filename
```

```
    git_cmd.show.split("\n")
```

```
  else
```

```
    File.read(filename).split("\n")
```

```
  end
```

```
end
```

```
end
```

class Listing

```
attr_reader :filename, :repository, :tag, :git_cmd
```

```
def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
```

```
  @filename = filename
```

```
  @repository = repository
```

```
  @tag = tag
```

```
  @git_cmd = git_cmd
```

```
end
```

```
def lines
```

```
  if git_cmd
```

```
    git_cmd.repository = repository
```

```
    git_cmd.tagname = tag
```

```
    git_cmd.filename = filename
```

```
    git_cmd.show.split("\n")
```

```
  else
```

```
    File.read(filename).split("\n")
```

```
  end
```

```
end
```

```
end
```



```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename      = filename
    @repository     = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname     = tag
      git_cmd.filename    = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end
end
```

Change #2

Change #2

Partial listings

Change #2

Partial listings

"1, 3-4, 15, 37-50"

*Easy
is the enemy of
simple*


```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd
  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename      = filename
    @repository     = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end
  # ...
end
```

```
class Listing
  attr_reader :filename,           :repository, :tag, :git_cmd
  def initialize(filename:,        repository: nil, tag: nil,
git_cmd: nil)
    @filename      = filename
    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end
  # ...
end
```

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil,
git_cmd: nil)
    @filename      = filename
    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end
  # ...
end
```

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil,
git_cmd: nil)
    @filename      = filename

    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end
  # ...
end
```

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil,
git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end
  # ...
end
```



```
class Listing
  # ...
```

```
class Listing
  # ...

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname     = tag
      git_cmd.filename    = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end
```

```
class Listing
  # ...

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname     = tag
      git_cmd.filename    = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end
```

 New Conditional Here

```
class Listing
  # ...

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname     = tag
      git_cmd.filename    = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end
```

```
class Listing
```

```
# ...
```

```
def lines
```

```
  if git_cmd
```

```
    git_cmd.repository = repository
```

```
    git_cmd.tagname     = tag
```

```
    git_cmd.filename    = filename
```

```
    git_cmd.show.split("\n")
```

```
  else
```

```
    File.read(filename).split("\n")
```

```
  end
```

```
end
```

```
def git_lines
```

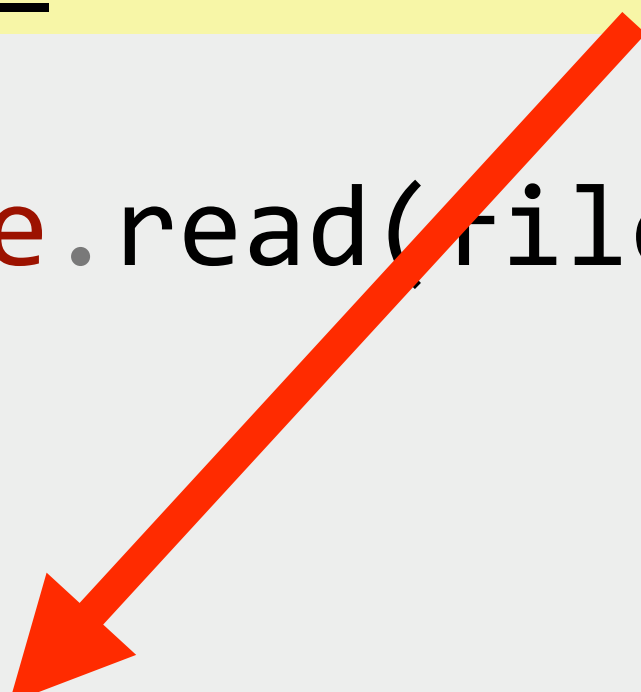
```
  git_cmd.repository = repository
```

```
  git_cmd.tagname     = tag
```

```
  git_cmd.filename    = filename
```

```
  git_cmd.show.split("\n")
```

```
end
```




```
class Listing
  # ...

  def lines
    if git_cmd
      git_lines

    else
      File.read(filename).split("\n")
    end
  end

  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname     = tag
    git_cmd.filename    = filename
    git_cmd.show.split("\n")
  end
end
```




```
class Listing
  # ...

  def lines
    if git_cmd
      git_lines

    else
      File.read(filename).split("\n")
    end
  end
end
```

```
class Listing
  # ...

  def lines
    if git_cmd
      git_lines
    else
      File.read(filename).split("\n")
    end
  end
end
```


```
class Listing
  # ...

  def lines
    if git_cmd
      git_lines
    else
      File.read(filename).split("\n")
    end
  end
end
```

```
class Listing
  # ...

  def lines
    if git_cmd
      git_lines
    else
      File.read(filename).split("\n")
    end
  end
end


def file_lines
  File.read(filename).split("\n")
end
```



```
class Listing
  # ...

  def lines
    if git_cmd
      git_lines
    else
      file_lines
    end
  end

  def file_lines
    File.read(filename).split("\n")
  end
end
```



```
class Listing
  # ...

  def lines
    if git_cmd
      git_lines
    else
      file_lines
    end
  end
end
```



```
class Listing
  # ...

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
  end
end
```

```
class Listing
  # ...

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
  end

end
```

```
class Listing
  # ...

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    if line_numbers
      return lines_to_print(all_lines)
    end

    all_lines
  end
end
```

```
class Listing
  # ...

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    if line_numbers
      return lines_to_print(all_lines)
    end

    all_lines
  end
end
```

```
# "1, 3-4, 15, 37-50"
```

```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
```

```
  specs.collect {|spec|
```

```
    edges = spec.split('-').collect(&:to_i)
```

```
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
```

```
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
```

```
  }.flatten.compact
```

```
end
```

```
# "1, 3-4, 15, 37-50"
```

```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
```

```
  specs.collect {|spec|
```

```
    edges = spec.split('-').collect(&:to_i)
```

```
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
```

```
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
```

```
  }.flatten.compact
```

```
end
```


"1, 3-4, 15, 37-50"



```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
```

```
  specs.collect {|spec|
```

```
    edges = spec.split('-').collect(&:to_i)
```

```
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
```

```
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
```

```
  }.flatten.compact
```

```
end
```

```
# "1, 3-4, 15, 37-50"
```

```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
```

```
  specs.collect {|spec|
```

```
    edges = spec.split('-').collect(&:to_i)
```

```
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
```

```
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
```

```
  }.flatten.compact
```

```
end
```

```
# "1, 3-4, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|
    edges = spec.split('-').collect(&:to_i)
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
  }.flatten.compact
end
```

```
# "1, 3-4, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|
    edges = spec.split('-').collect(&:to_i)
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
  }.flatten.compact
end
```

Ouch

```
# "1, 3-4, 15, 37-50"
```

```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
```

```
  specs.collect {|spec|
```

```
    edges = spec.split('-').collect(&:to_i)
```

```
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
```

```
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
```

```
  }.flatten.compact
```

```
end
```

Change #3

Change #3

Dynamic Comments

Change #3

Dynamic Comments

"1, 3-4, 15, 37-50"

Change #3

Dynamic Comments

"1, 3-4, 15, 37-50"

Change #3

Dynamic Comments

"1, #4, 3-4, #6, 15, 37-50"

```
# "1, 3-4, 15, 37-50"
```

```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
```

```
  specs.collect {|spec|
```

```
    edges = spec.split('-').collect(&:to_i)
```

```
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
```

```
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
```

```
  }.flatten.compact
```

```
end
```

```
# "1, #4, 3-4, #6, 15, 37-50"
```

```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
```

```
  specs.collect {|spec|
```

```
    edges = spec.split('-').collect(&:to_i)
```

```
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
```

```
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
```

```
  }.flatten.compact
```

```
end
```

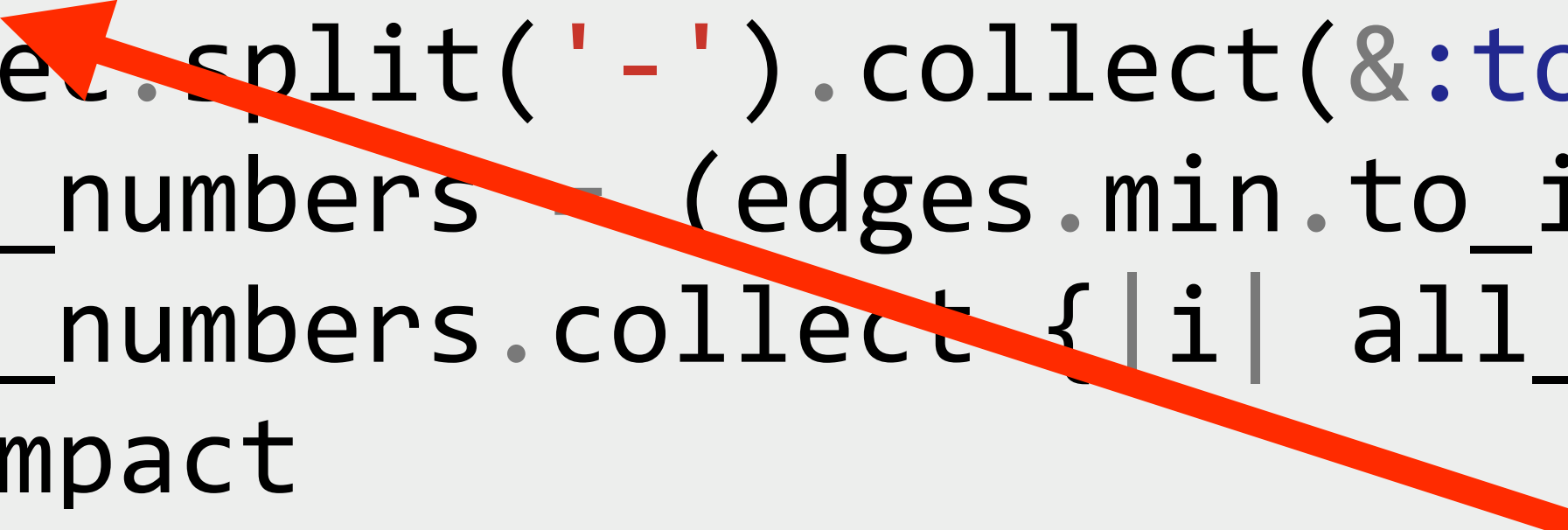
```
# "1, #4, 3-4, #6, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|
    edges = spec.split('-').collect(&:to_i)
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
  }.flatten.compact
end
```



```
# "1, #4, 3-4, #6, 15, 37-50"
```

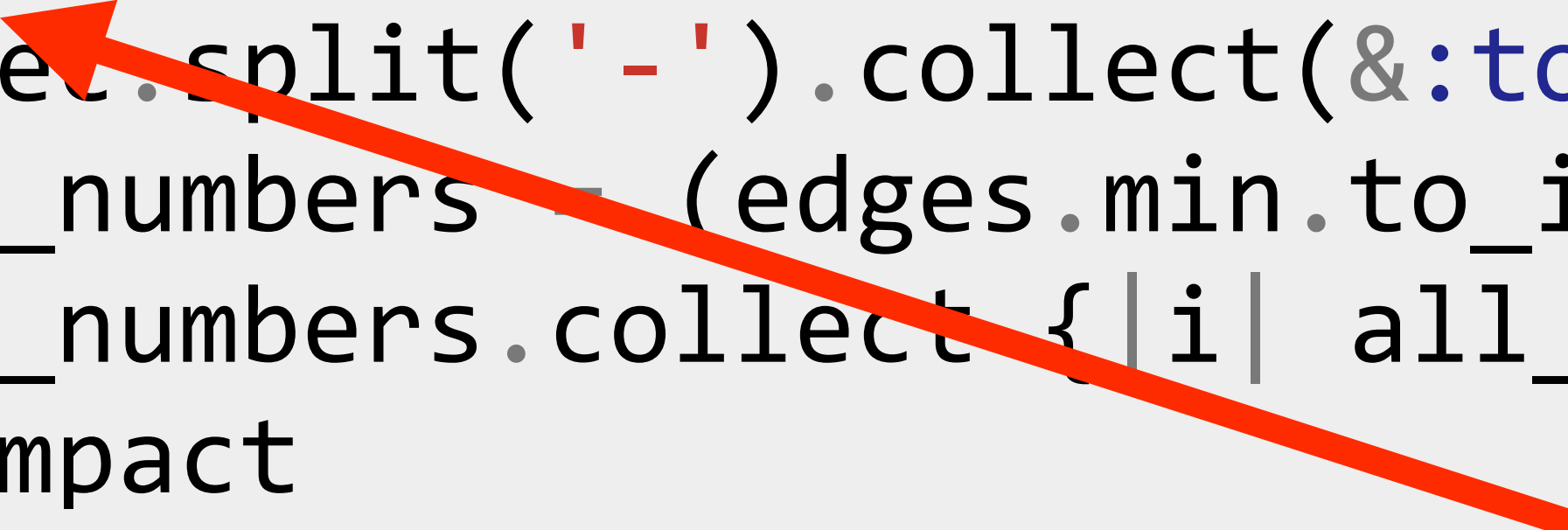
```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|
    edges = spec.split('-').collect(&:to_i)
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
  }.flatten.compact
end
```



New Conditional Here

```
# "1, #4, 3-4, #6, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|
    edges = spec.split('-').collect(&:to_i)
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
  }.flatten.compact
end
```



New Conditional Here

```
# "1, #4, 3-4, #6, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|

    edges = spec.split('-').collect(&:to_i)
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
    individual_numbers.collect {|i| all_lines[i - 1]}.compact
  }.flatten.compact
end
```

```
# "1, #4, 3-4, #6, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|

    edges = spec.split('-').collect(&:to_i)
    individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
    individual_numbers.collect {|i| all_lines[i - 1]}.compact

  }.flatten.compact
end
```

```
# "1, #4, 3-4, #6, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|
    if spec.include?('#')

    else
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    end
  }.flatten.compact
end
```

```
# "1, #4, 3-4, #6, 15, 37-50"
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
  specs.collect {|spec|
    if spec.include?('#')
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    else
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    end
  }.flatten.compact
end
```

Change #4

Change #4

Left Justification

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil,
git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end
  # ...
```

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil,
repository: nil, tag: nil, git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @repository     = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  # ...
```

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, left_justify: false,
repository: nil, tag: nil, git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @repository     = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  # ...
```

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, left_justify: false,
repository: nil, tag: nil, git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers

    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  # ...
```

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, left_justify: false,
repository: nil, tag: nil, git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @left_just     = left_justify
    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  # ...
```

```
class Listing
  # ...
```



```
class Listing
  # ...

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    if line_numbers
      return lines_to_print(all_lines)
    end


    all_lines
  end
end
```

```
class Listing
  # ...

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    if line_numbers
      return lines_to_print(all_lines)
    end

    all_lines
  end
end
```



Yup

```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

  if line_numbers
    return lines_to_print(all_lines)
  end

  all_lines
end
```

```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

  if line_numbers
    return lines_to_print(all_lines)
  end

  all_lines
end
```

```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

  if line_numbers
    return lines_to_print(all_lines)
  end

  all_lines
end
```

```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

    if line_numbers
      lines_to_print(all_lines)
    else
      all_lines
    end
end
```

```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

  subset =
    if line_numbers
      lines_to_print(all_lines)
    else
      all_lines
    end
end
```



```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end
  subset =
    if line_numbers
      lines_to_print(all_lines)
    else
      all_lines
    end
end
```

```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

  subset =
    if line_numbers
      lines_to_print(all_lines)
    else
      all_lines
    end

  if left_just
    return justify(subset)
  end
  subset
end
```

```
def lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

  subset =
    if line_numbers
      lines_to_print(all_lines)
    else
      all_lines
    end

  if left_just
    return justify(subset)
  end
  subset
end
```

```
def justify(lines)
  lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || ""}
end

def num_leading_spaces_to_remove(lines)
  @num ||=
    lines.reduce(999999) {|current_min, line|
      line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
    }
end

def num_leading_spaces(line)
  line[/\A */].size
end
```

```
def justify(lines)
  lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || ""}
end

def num_leading_spaces_to_remove(lines)
  @num ||=
    lines.reduce(999999) {|current_min, line|
      line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
    }
end

def num_leading_spaces(line)
  line[/\A */].size
end
```

```
def justify(lines)
  lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || ""}
end

def num_leading_spaces_to_remove(lines)
  @num ||=
    lines.reduce(999999) {|current_min, line|
      line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
    }
end

def num_leading_spaces(line)
  line[/\A */].size
end
```

```
def justify(lines)
  lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || ""}
end

def num_leading_spaces_to_remove(lines)
  @num ||=
    lines.reduce(999999) {|current_min, line|
      line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
    }
end

def num_leading_spaces(line)
  line[/\A */].size
end
```

Ouch

Progression



1 File *or* Git Tag

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir="#{repository}"]
  end
end
```

Lines: 29

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd
  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir="#{repository}"]
  end
end
```

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd
  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir="#{repository}"]
  end
end
```

Execution Paths: 2

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  tr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

2 All *or* Some Lines

2

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    if line_numbers
      return lines_to_print(all_lines)
    end

    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")

    specs.collect {|spec|
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

Lines: 50

2

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    if line_numbers
      return lines_to_print(all_lines)
    end

    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/[' ']/, "").gsub(/ /, '').split(",")

    specs.collect {|spec|
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

Execution Paths:

2

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end
  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

Execution Paths: 4

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

2

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['?]/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

3 Code Line *or* Comment

3

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @repository     = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    if line_numbers
      return lines_to_print(all_lines)
    end

    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname     = tag
    git_cmd.filename    = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")

    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

Lines: 55

3

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end
  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

Execution Paths:

3

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end
  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ' ').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

Execution Paths: 8

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

2

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

3

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

4 Raw *or* Left Just

4

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
  private
  #####
  # Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  # Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?(' ')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  # Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/^\A */].size
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ lines of error handling
```

Lines: 83

4

Execution Paths:

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  f lines
  all_lines =
    if git_cmd
      git_lines
    else
      file_lines
    end

  subset =
    if line_numbers
      lines_to_print(all_lines)
    else
      all_lines
    end

  if left_just
    return justify(subset)
  end
  subset
end

private
#####
Reading
#####
f git_lines
git_cmd.repository = repository
git_cmd.tagname = tag
git_cmd.filename = filename
git_cmd.show.split("\n")
end

f file_lines
File.read(filename).split("\n")
end

#####
Subsetting
#####
f lines_to_print(all_lines)
specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")

specs.collect {|spec|
  if spec.include?('#')
    num_spaces = spec.delete("#").to_i
    (" " * num_spaces) + "# ..."
  else
    edges = spec.split('-').collect(&:to_i)
    range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
    range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
  end
}.flatten.compact
end

#####
Justification
#####
f justify(lines)
lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
end

f num_leading_spaces_to_remove(lines)
@num ||=
  lines.reduce(999999) {|current_min, line|
    line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
  }
end

f num_leading_spaces(line)
line[/\A */].size
end

end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  f show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  f git_dir
    %Q[--git-dir=#{repository}]
  end
end

end
# plus 90+ lines of error handling
```

4

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end

  private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end

  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end

  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end

  def num_leading_spaces(line)
    line[/\A */].size
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end

# plus 90+ lines of error handling
```

Execution Paths: 16

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

2

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

3

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?("#")
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

4

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, left_just: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_just
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end

  private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?("#")
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end

  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end

  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end

  def num_leading_spaces(line)
    line[/\A */].size
  end
end

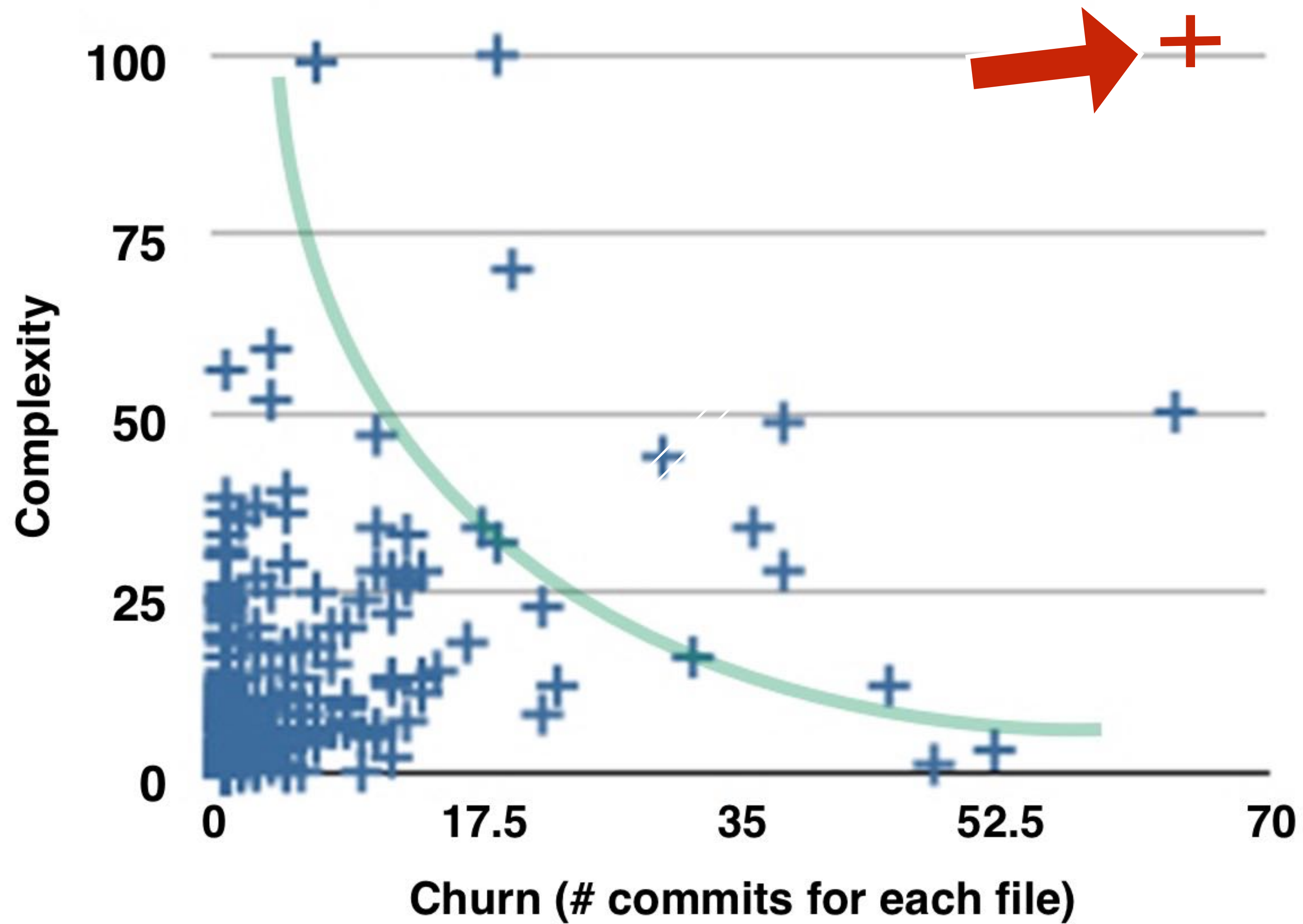
class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end

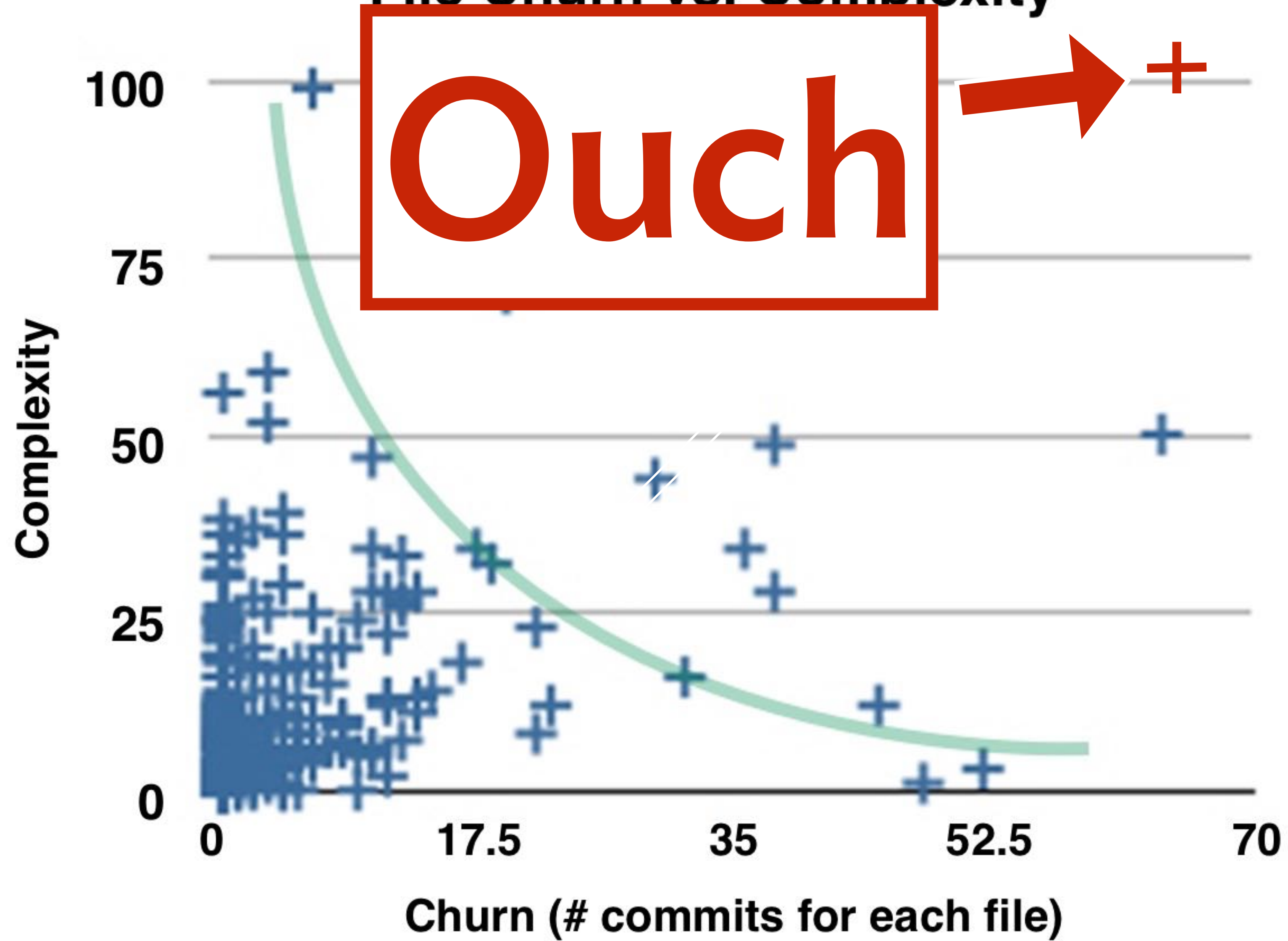
  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end

# plus 90+ lines of error handling
```

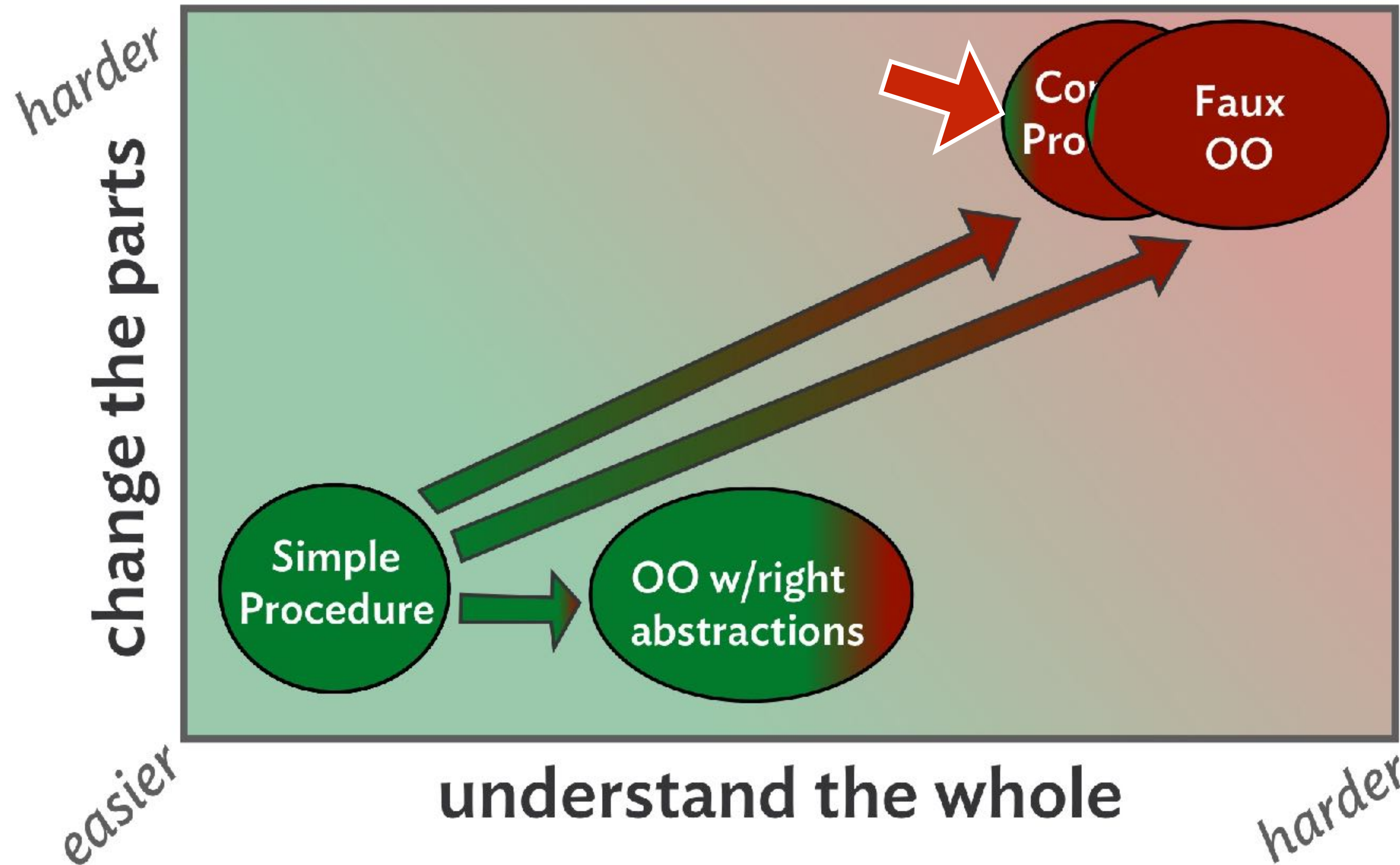
File Churn vs. Complexity



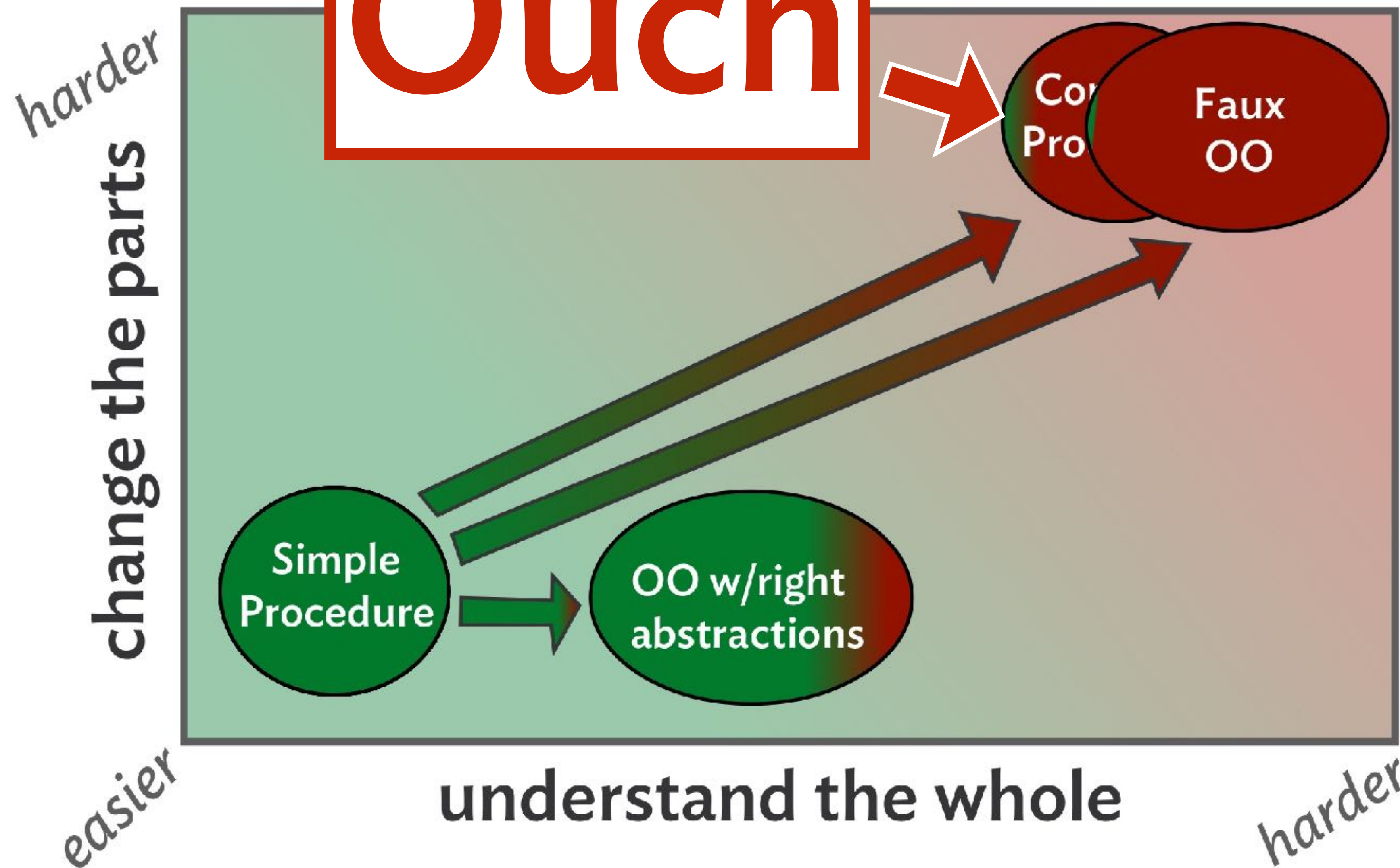
File Churn vs. Complexity

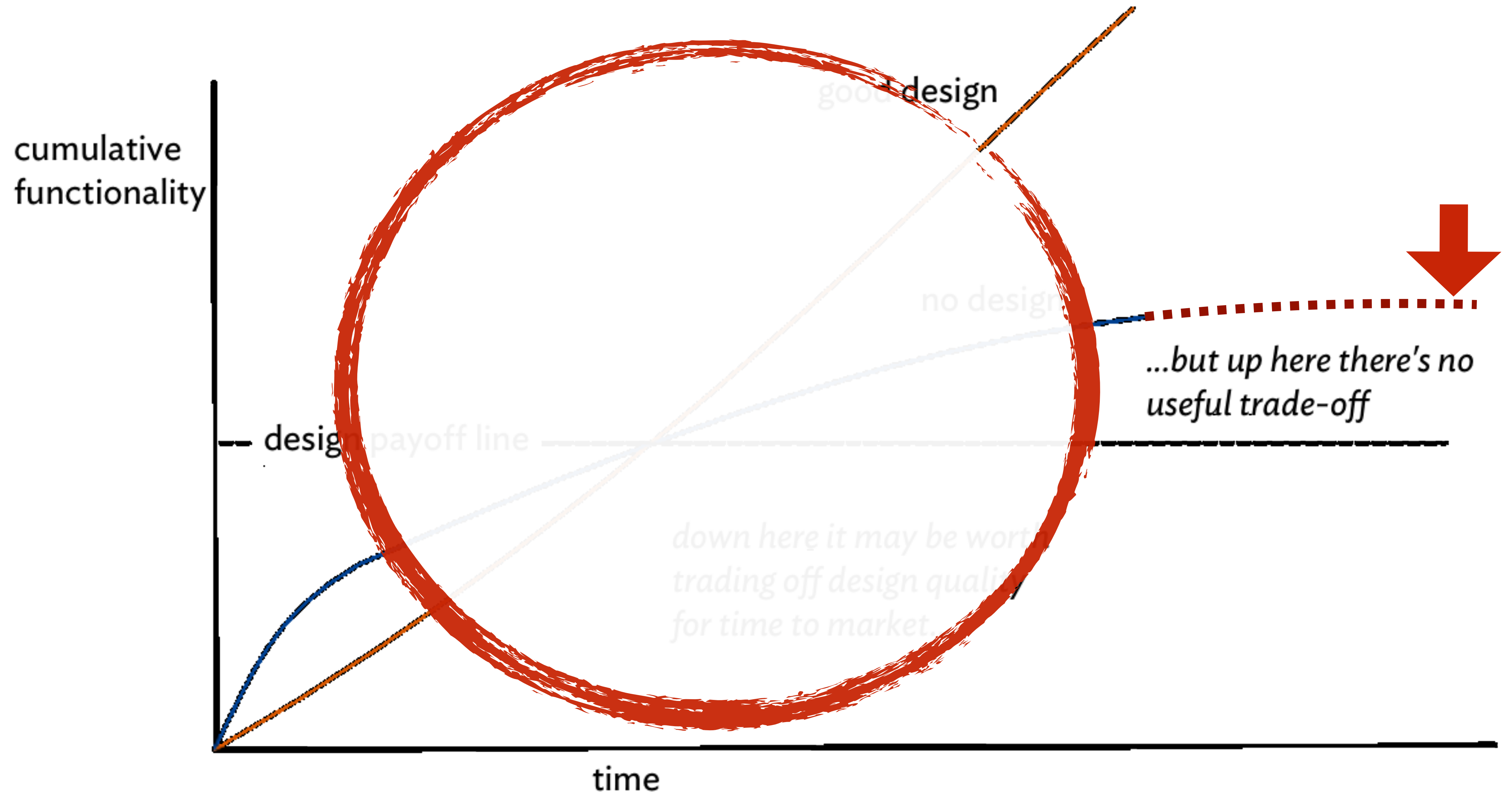


Procedural vs Object-Oriented Code

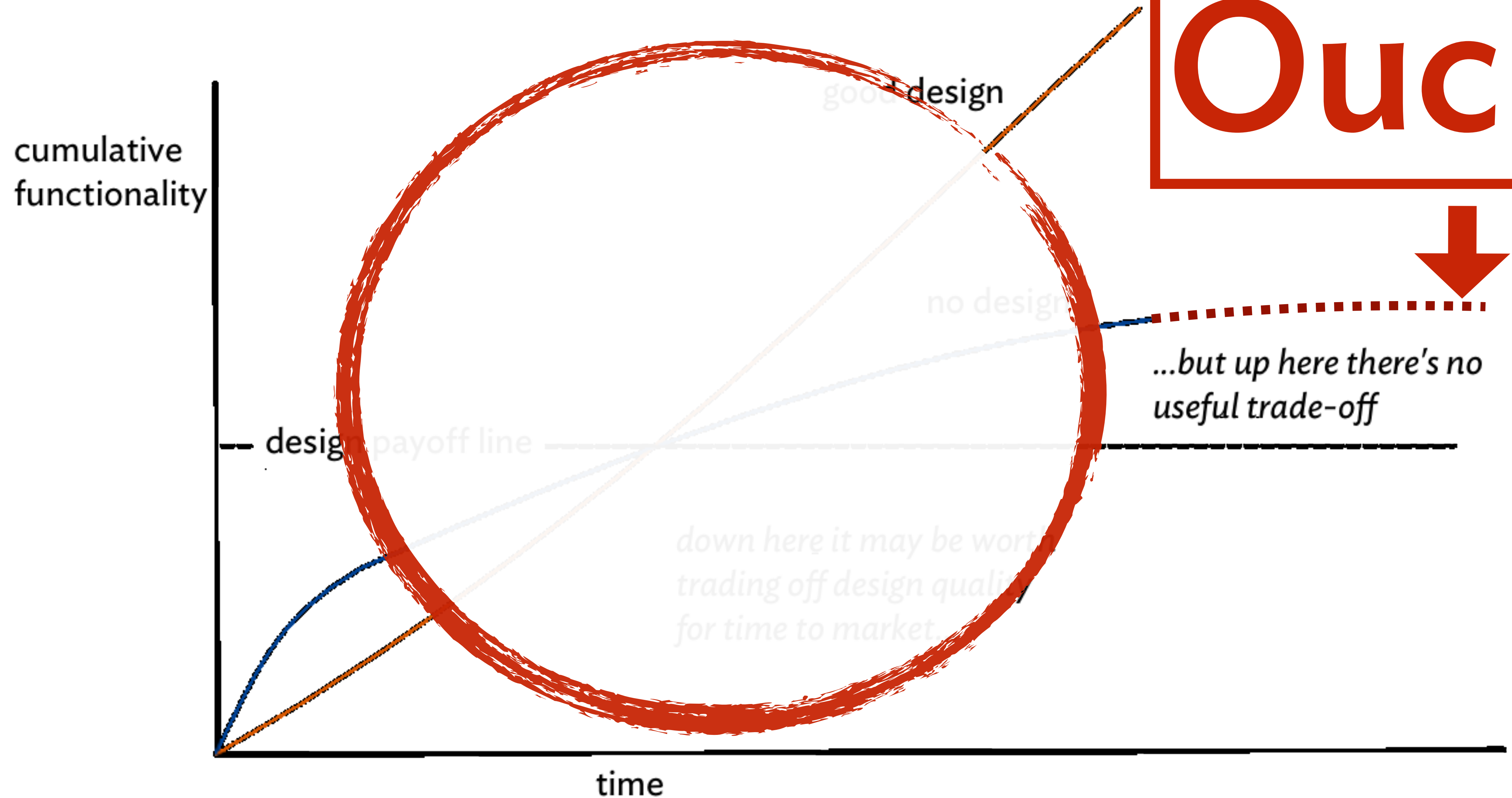


Process of Writing Object-Oriented Code





Ouch



What brought you success

*What brought you success
will doom you to failure.*

Affordances



"Doorknob" by [photonooner](#) CC BY-NC-ND



"Doorknob" by [photonooner](#) CC BY-NC-ND



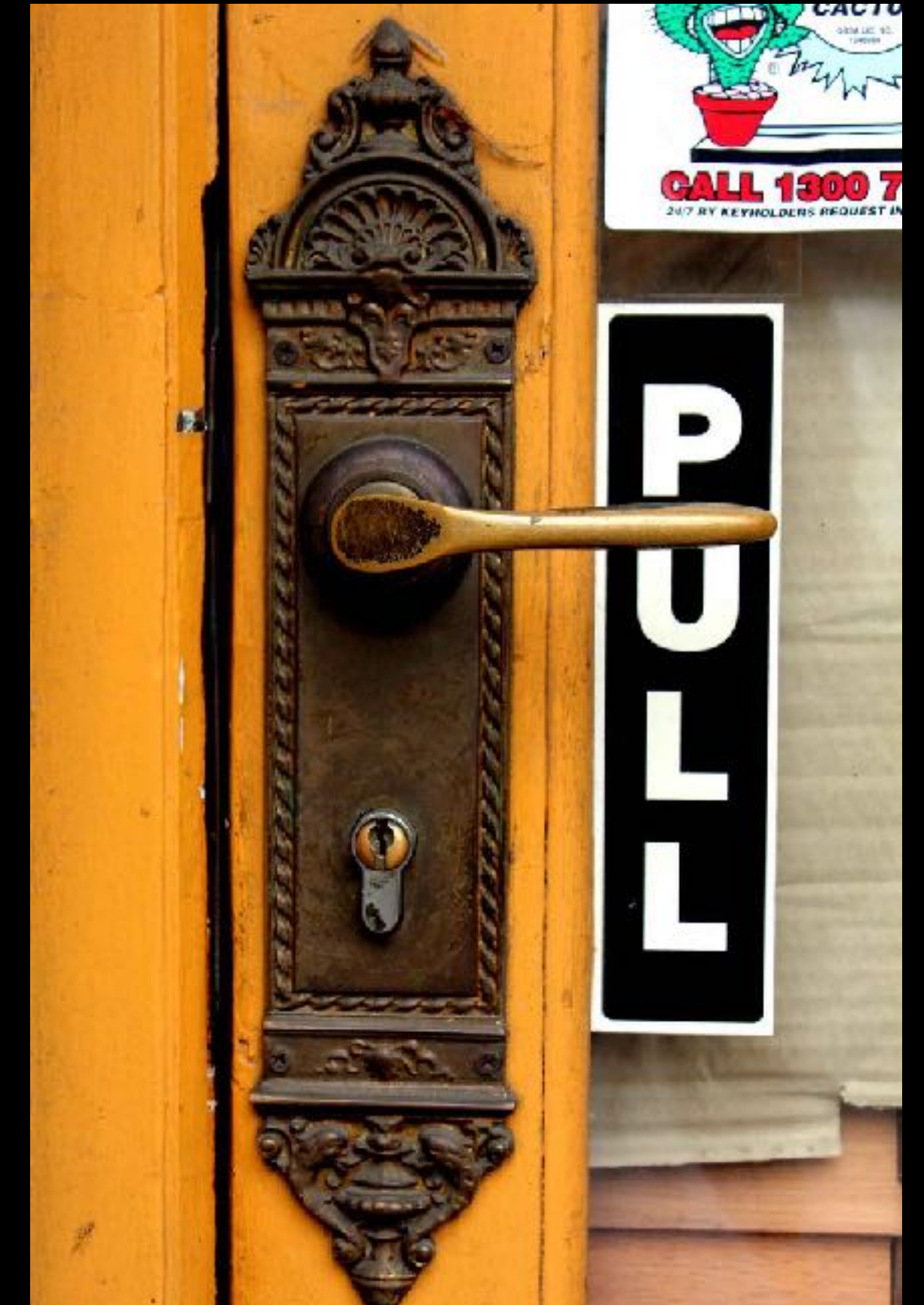
"Door Handle" by [www.trek.today](#) CC BY-NC-ND



"Doorknob" by [photonooner](#) CC BY-NC-ND



"Door Handle" by [www.trek.today](#) CC BY-NC-ND



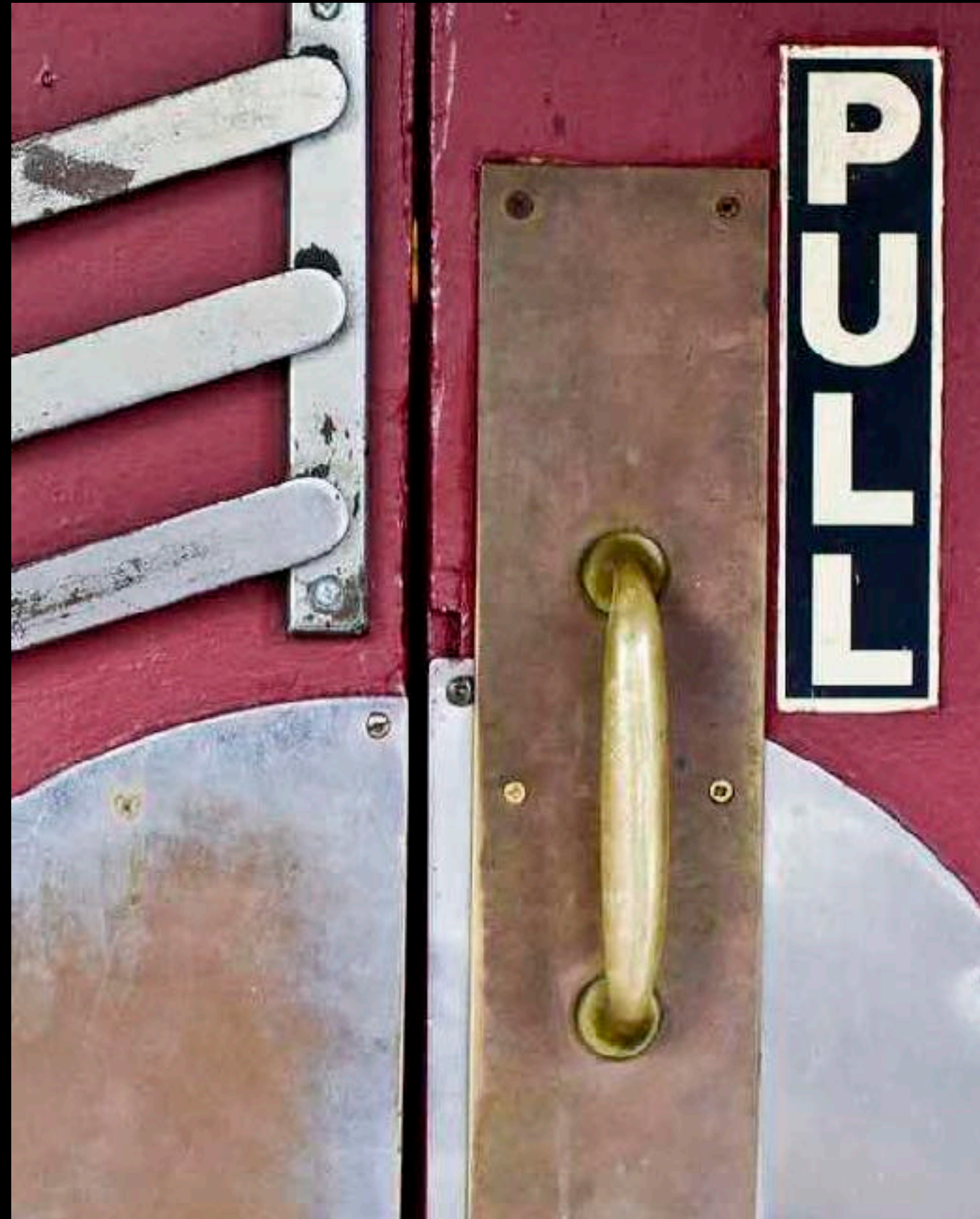
"pull" by [various brennemans](#) CC BY-SA



"Untitled" by [vibeke_fn](#) CC BY-NC-ND



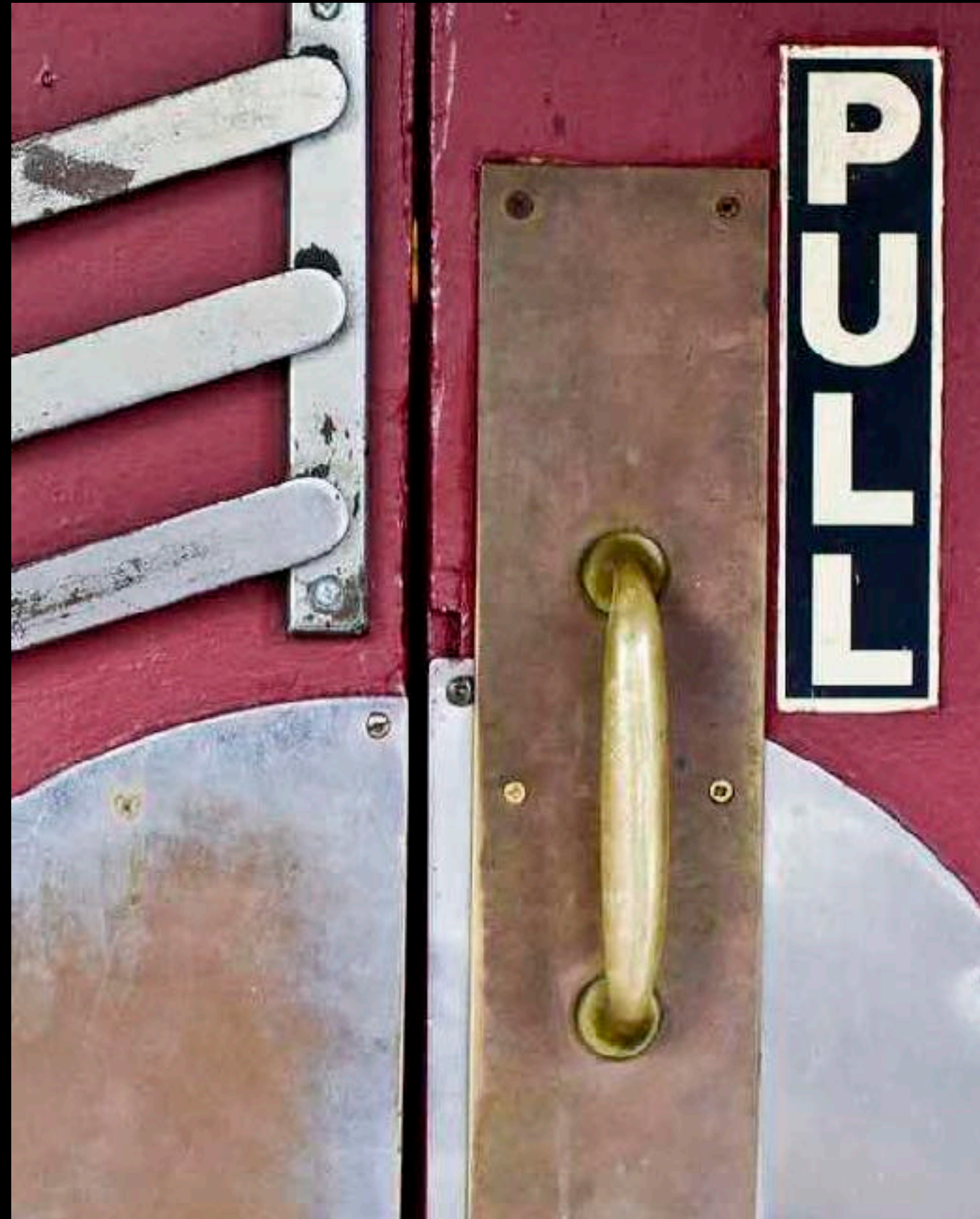
"Untitled" by [vibeke_fn](#) CC BY-NC-ND



"pull" by [greenkozi](#) CC BY-NC-ND



"Untitled" by [vibeke_fn](#) CC BY-NC-ND



"pull" by [greenkozi](#) CC BY-NC-ND



"door push plate" by [stu_spivack](#) CC BY-SA

Oo
Affords

Oo

Affords

Anthropomorphic

OO
Affords

Anthropomorphic

Polymorphic

OO
Affords

Anthropomorphic

Polymorphic

Loosely-coupled

OO
Affords

Anthropomorphic

Polymorphic

Loosely-coupled

Role-playing

OO
Affords

Anthropomorphic

Polymorphic

Loosely-coupled

Role-playing

Factory-created

OO *Affords*

Anthropomorphic

Polymorphic

Loosely-coupled

Role-playing

Factory-created

Message-sending

OO
Affords

Anthropomorphic

Polymorphic

Loosely-coupled

Role-playing

Factory-created

Message-sending

Objects

Anthropomorphism

Anthropomorphism

-the attribution of human traits, emotions or intentions to non-human entities.



DSCF2551 by clownhousethethird CC

Polymorphism

Polymorphism

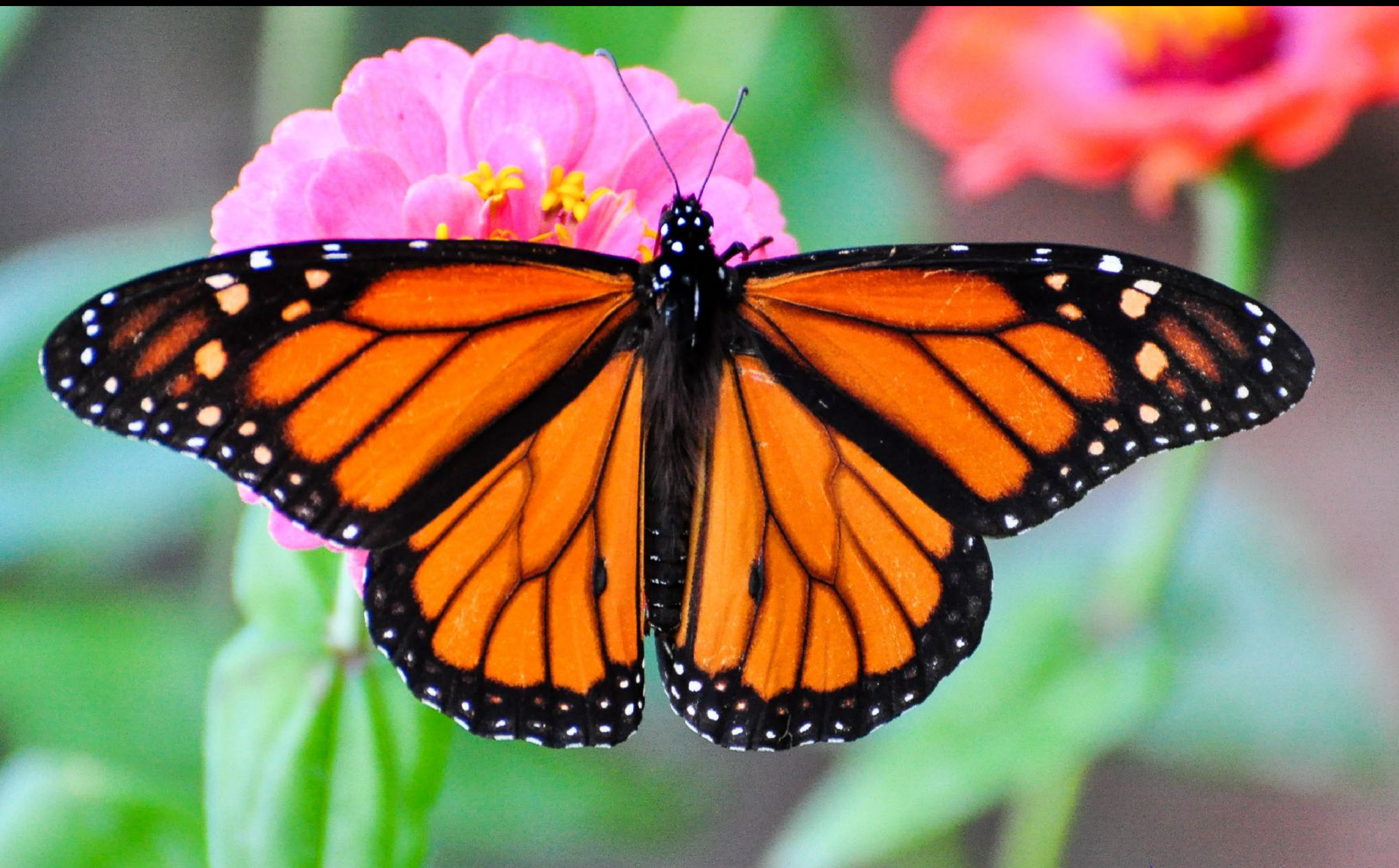
*-the condition of occurring in
several different forms*



DSCF2551 by clownhousethethird CC



Viceroy 001 by Lynnea A. Parker Photography CC BY-NC



Loosely-Coupled

Loosely-Coupled

-objects strive for independence



Gweck Tug of War by [studiostein](#) CC BY-NC

Role-playing

Role-playing

*-objects are more players of their roles
than instances of their types*



2017 WVB | Club Teams 14U by Sangudo CC BY-NC-ND

Factory-created

Factory-created

*-factories hide the rules for picking
the right player of a role*



tokuyama factory chimney by [Kenichi Nobusue](#) CC BY-NC-ND



_IMG7252.jpg by NicolasRuh CC BY-ND

Message-sending

Message-sending

*-I know what I want,
you know how to do it*



See No Evil, Hear No Evil, Speak No Evil by PMillera4 CC BY-NC-ND

OO
Affords

Anthropomorphic

Polymorphic

Loosely-coupled

Role-playing

Factory-created

Message-sending

Objects

Resolution

Resolution

Where all ends well

```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

2

```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

2X2

```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```


2X2X2

```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

2X2X2

```
class Listing
```

```
  def lines
```

```
    all_lines =
```

```
    if git_cmd
      git_lines
```

```
    else
      file_lines
```

```
    end
```

```
    subset =
```

```
    if line_numbers
      lines_to_print(all_lines)
```

```
    else
      all_lines
```

```
    end
```

```
  if left_just
```

```
    return justify(subset)
```

```
  end
```

```
  subset
```

```
end
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/['|']/,
```

```
  specs.collect {|spec|
```

```
    if spec.include?('#')
      num_spaces = spec.delete("#").
        (" " * num_spaces) + "# ..."
```

```
    else
      edges = spec.split('-').collect
      range_of_line_numbers = (edges
        range_of_line_numbers.collect
```

```
    end
```

```
  }.flatten.compact
```

```
end
```

2X2X2X2

```
class Listing
```

```
  def lines
```

```
    all_lines =
```

```
    if git_cmd
      git_lines
```

```
    else
```

```
      file_lines
```

```
    end
```

```
    subset =
```

```
    if line_numbers
      lines_to_print(all_lines)
```

```
    else
```

```
      all_lines
```

```
    end
```

```
    if left_just
```

```
      return justify(subset)
```

```
    end
```

```
    subset
```

```
  end
```

```
  def lines_to_print(all_lines)
```

```
    specs = line_numbers.gsub(/['|']/,
```

```
    specs.collect {|spec|
```

```
      if spec.include?('#')
```

```
        num_spaces = spec.delete("#").
```

```
        (" " * num_spaces) + "# ..."
```

```
      else
```

```
        edges = spec.split('-').collect
```

```
        range_of_line_numbers = (edges
```

```
        range_of_line_numbers.collect
```

```
      end
```

```
    }.flatten.compact
```

```
  end
```



```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
end
```

If you know enough
to inject this




```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
end
```

Inject a smarter thing

```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
end
```

1 File *or* Git Tag

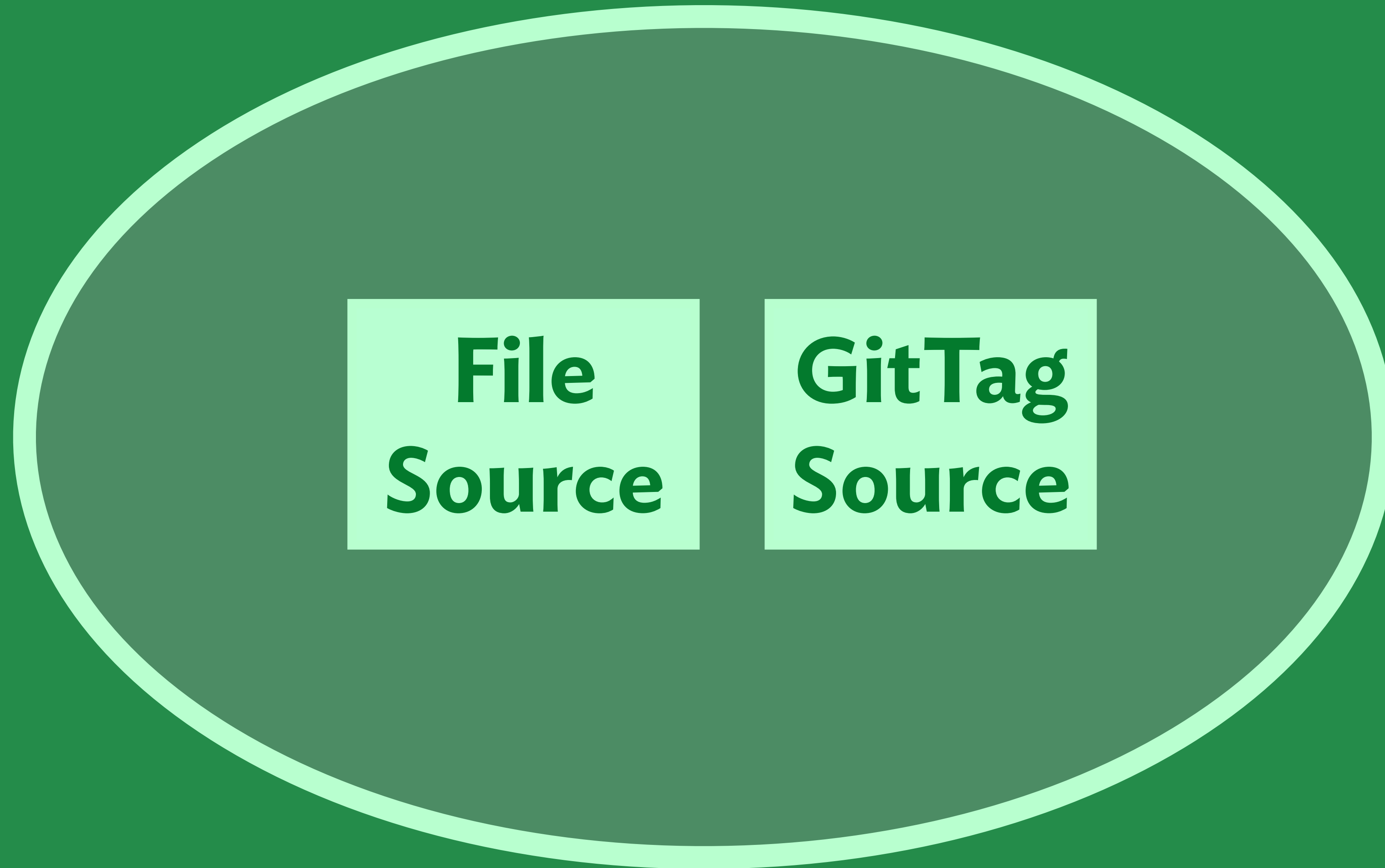
```
class Listing
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
end
```

1 Role = Source

Source

Source #lines

Source #lines



SOURCES

```
class FileSource  
end
```

```
class GitTagSource  
end
```

```
class FileSource
```

```
end
```

```
class GitTagSource
```

```
end
```

```
class FileSource
  def lines

    end
    # ...
end
```

```
class GitTagSource
  def lines

    end
    # ...
end
```

```
class FileSource
  def lines

    end
    # ...
end
```

```
class GitTagSource
  def lines

    end
    # ...
end
```



```
class FileSource
  def lines

  end
  # ...
end
```

```
class GitTagSource
  def lines

  end
  # ...
end
```

Polymorphism

```
class FileSource
  def lines
    File.read(filename).split("\n")
  end
  # ...
end

class GitTagSource
  def lines

  end
  # ...
end
```

```
class FileSource
  def lines
    File.read(filename).split("\n")
  end
  # ...
end

class GitTagSource
  def lines
    git_cmd.repository = repository
    git_cmd.tagname     = tag
    git_cmd.filename    = filename
    git_cmd.show.split("\n")
  end
  # ...
end
```

*Isolate the things
you want to vary*

```

class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil,
git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @left_just     = left_justify
    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end

  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname     = tag
    git_cmd.filename    = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end
end

```



```

class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil,
git_cmd: nil)
    @filename      = filename
    @line_numbers  = line_numbers
    @left_just     = left_justify
    @repository    = repository
    @tag           = tag
    @git_cmd       = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end

```

```

def git_lines
  git_cmd.repository = repository
  git_cmd.tagname     = tag
  git_cmd.filename    = filename
  git_cmd.show.split("\n")
end

```

```

def file_lines
  File.read(filename).split("\n")
end

```

```
class Listing
  attr_reader :line_numbers, :left_just

  def initialize(
    line_numbers: nil, left_justify: false)

    @line_numbers = line_numbers
    @left_just    = left_justify

  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```

```

class Listing
  attr_reader :line_numbers, :left_just

  def initialize(
    line_numbers: nil, left_justify: false)

    @line_numbers = line_numbers
    @left_just    = left_justify

  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end

```

```
class Listing
  attr_reader :line_numbers, :left_just

  def initialize(
    line_numbers: nil, left_justify: false)

    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```


```
class Listing
  attr_reader      :line_numbers, :left_just

  def initialize(
    line_numbers: nil, left_justify: false)

    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```

 Inject a Source


```
class Listing
  attr_reader :line_numbers, :left_just

  def initialize(
    line_numbers: nil, left_justify: false)

    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```

```
class Listing
  attr_reader :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)

    @line_numbers = line_numbers
    @left_just     = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```

```
class Listing
  attr_reader      :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source        = source
    @line_numbers  = line_numbers
    @left_just     = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```



```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end

    subset = # ...
  end
end
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end


  def lines
    all_lines = source.lines

    subset = # ...
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
```



```
subset = # ...
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end
```

```
  def lines
    all_lines source.lines
```

```
    subset = # ...
```

Execution
Paths: **16?**

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end
```

```
  def lines
    all_lines source.lines
```

```
    subset = # ...
```

Execution
Paths: ↓8

*Push conditionals
back on the stack*

```
class Listing
  attr_reader :source, :line_numbers, :left_just
  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :line_numbers, :left_just
  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

2

All *or* Some Lines


```
class Listing
  attr_reader :source, :line_numbers, :left_just

  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

2

Subset

```
class Listing
  attr_reader :source, :line_numbers, :left_just
  def initialize(source:, line_numbers: nil, left_justify: false)
    @source      = source
    @line_numbers = line_numbers
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

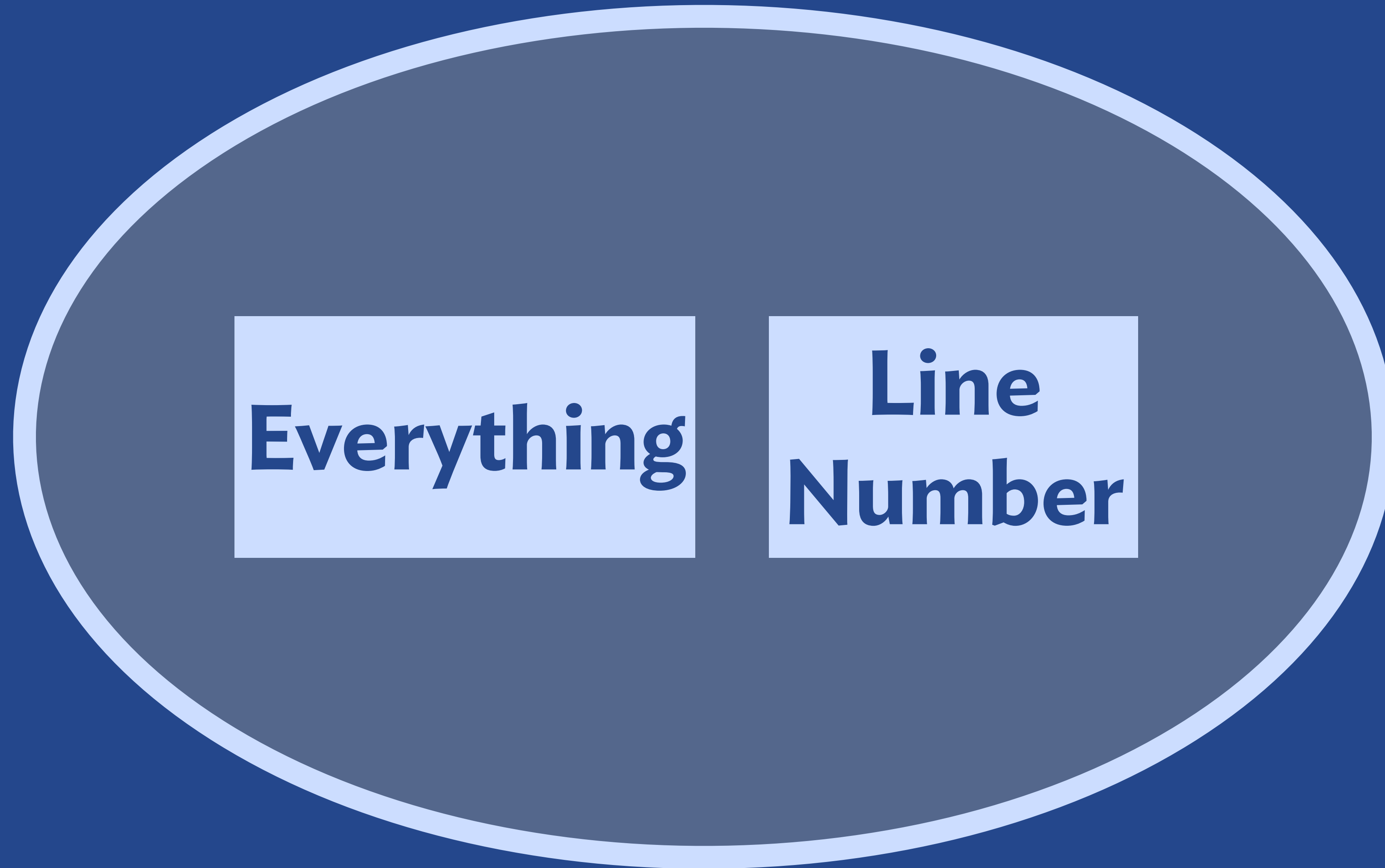


Inject a smarter thing

Subset

Subset #lines

Subset #lines



SUBSET

```
module Subset
  class Everything
  end

  class LineNumber
  end
end
```

SUBSET

```
module Subset  
  class Everything
```

```
end
```

```
class LineNumber
```

```
end
```

SUBSET

```
module Subset
  class Everything
    def lines(everything)

      end
    end

    class LineNumber

      end
  end
end
```

SUBSET

```
module Subset
  class Everything
    def lines(everything)
      everything
    end
  end

  class LineNumber

end
```

SUBSET

```
module Subset  
  class Everything
```

```
end
```

```
class LineNumber
```

```
end
```


SUBSET

@sandimetz

```
module Subset
  class Everything; end

  class LineNumber
```

end

SUBSET

```
module Subset
  class Everything; end

  class LineNumber
    # ...
    def lines_to_print(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, " ")
      specs.collect {|spec|
        if spec.include?('#')
          num_spaces = spec.delete("#").to_i
          (" " * num_spaces) + "# ..."
        else
          edges = spec.split('-').collect(&:to_i)
          range_of_line_numbers = (edges.min.to_i..edges.max.to_i)
          range_of_line_numbers.collect {|i| all_line_numbers[i]}
        end
      }.flatten.compact
    end
  end
end
```

SUBSET

```
module Subset
  class Everything; end

  class LineNumber
    # ...
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, " ")
      specs.collect {|spec|
        if spec.include?('#')
          num_spaces = spec.delete("#").to_i
          (" " * num_spaces) + "# ..."
        else
          edges = spec.split('-').collect(&:to_i)
          range_of_line_numbers = (edges.min.to_i..edges.max.to_i)
          range_of_line_numbers.collect {|i| all_line_numbers[i]}
        end
      }.flatten.compact
    end
  end
end
```

SUBSET

```
module Subset
  class Everything; end

  class LineNumber
    # ...
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, " ")
      specs.collect {|spec|
        if spec.include?('#')
          num_spaces = spec.delete("#").to_i
          (" " * num_spaces) + "# ..."
        else
          edges = spec.split('-').collect(&:to_i)
          range_of_line_numbers = (edges.min.to_i..edges.max.to_i)
          range_of_line_numbers.collect {|i| all_line_numbers[i]}
        end
      }.flatten.compact
    end
  end
end
```



class Listing

```
attr_reader :source, :line_numbers, :left_just
```

```
def initialize(source:, line_numbers: nil, left_justify: false)
```

```
  @source      = source
```

```
  @line_numbers = line_numbers
```

```
  @left_just    = left_justify
```

```
end
```

```
def lines
```

```
  all_lines = source.lines
```

```
  subset =
```

```
    if line_numbers
```

```
      lines_to_print(all_lines)
```

```
    else
```

```
      all_lines
```

```
    end
```

```
  if left_just
```

```
    return justify(subset)
```

```
  end
```

```
  subset
```

```
end
```

```
def lines_to_print(all_lines)
```

```
  specs = line_numbers.gsub(/
```

```
  specs.collect {|spec|
```

```
    if spec.include?('#')
```

```
      num_spaces = spec.delete
```

```
        (" " * num_spaces) + "#
```

```
    else
```

```
      edges = spec.split('-').
```

```
      range_of_line_numbers =
```

```
      range_of_line_numbers.co
```

```
    end
```

```
  }.flatten.compact
```

```
end
```


class Listing

```
attr_reader :source, :line_numbers, :left_just
```

```
def initialize(source:, line_numbers: nil, left_justify: false)
```

```
  @source = source
```

```
  @line_numbers = line_numbers
```

```
  @left_just = left_justify
```

```
end
```

```
def lines
```

```
  all_lines = source.lines
```

```
  subset =
```

```
    if line_numbers
      lines_to_print(all_lines)
```

```
    else
      all_lines
```

```
    end
```

```
  if left_just
```

```
    return justify(subset)
```

```
  end
```

```
  subset
```

```
end
```

```
def lines_to_print(all_lines)
  specs = line_numbers.gsub(/[
  specs.collect {|spec|
    if spec.include?('#')
      num_spaces = spec.delete(
        " " * num_spaces) + "#"
    else
      edges = spec.split('-').
      range_of_line_numbers =
      range_of_line_numbers.co
    end
  }.flatten.compact
end
```

```

class Listing
  attr_reader :source,                :left_just
  def initialize(source:,              left_justify: false)
    @source      = source

    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end

```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source = source

    @left_just = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just

  def initialize(source:, subsetter:, left_justify: false)
    @source      = source

    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just

  def initialize(source:, subsetter:, left_justify: false)
    @source      = source

    @left_just = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```



```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset = 
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end


  def lines
    all_lines = source.lines
    subset = subsetter.lines(all_lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end

  def lines
    all_lines = source.lines
    subset = subsetter.lines(all_lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

A red arrow originates from the 'lines' attribute access in the line 'subset = subsetter.lines(all_lines)' and points diagonally upwards and to the left towards the '@subsetter' class attribute definition in the 'initialize' method.

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end

  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```



```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

Execution
Paths: ↓4

*Dependency injection
is your friend*

*As long as
you inject smart things*

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```

4 Raw *or* Left Just


```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end
  def lines
    subset = subsetter.lines(source.lines)

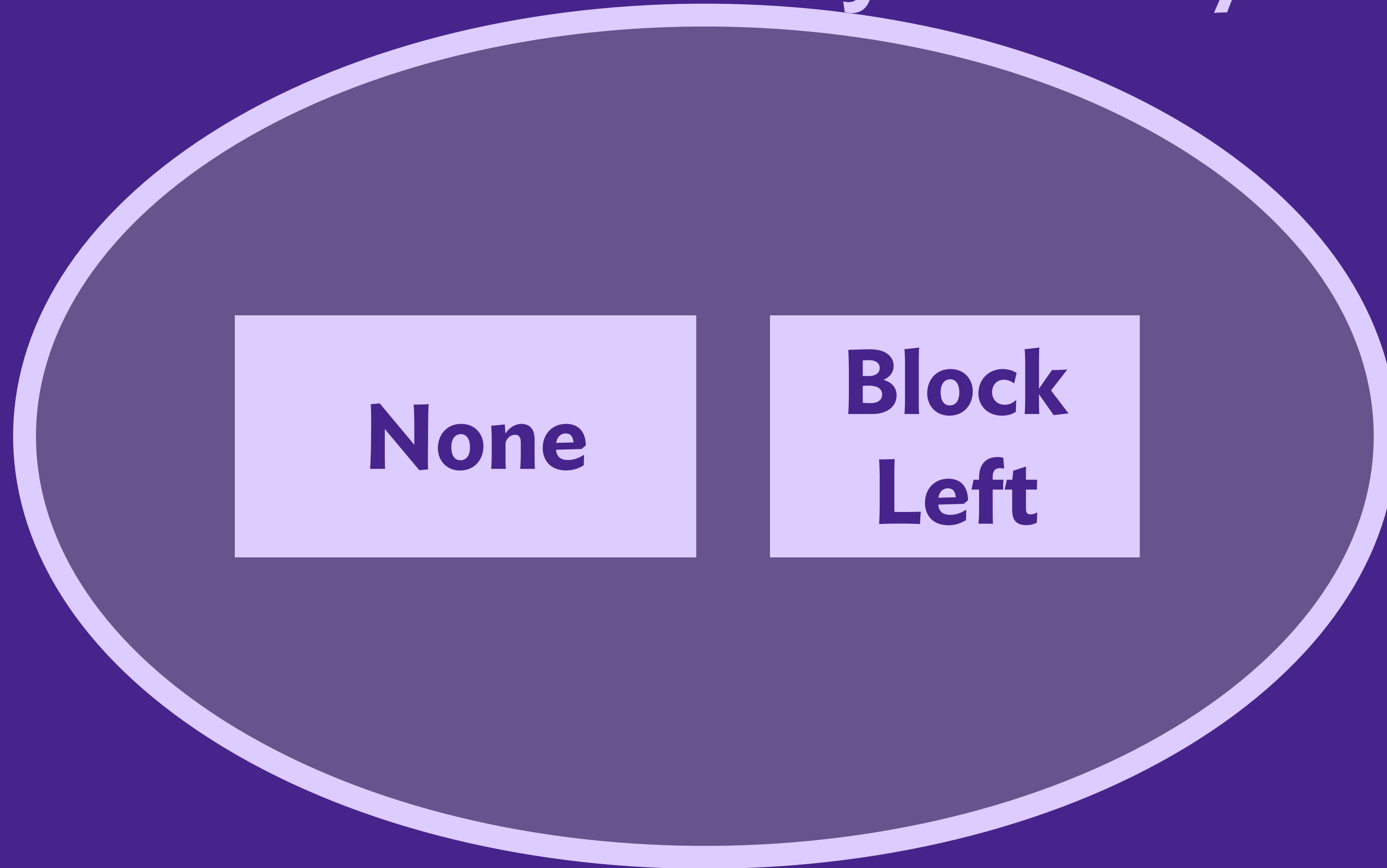
    if left_just
      return justify(subset)
    end
    subset
  end
end
```

4 Justifier

Justifier

Justifier #justify

Justifier #justify



```
module Justification
  class None
    def justify(lines)
      lines
    end
  end

  class BlockLeft
    def justify(lines)
      lines.map {|line| line.slice(num_leading_spaces)}
    end

    def num_leading_spaces_to_remove; end
    def num_leading_spaces(line); end
  end
end
```

```
module Justification
  class None
    def justify(lines)
      lines
    end
  end

  class BlockLeft
    def justify(lines)
      lines.map {|line| line.slice(num_leading_spaces)}
    end

    def num_leading_spaces_to_remove; end

    def num_leading_spaces(line); end
  end
end
```



```
module Justification
  class None
    def justify(lines)
      lines
    end
  end

  class BlockLeft
    def justify(lines)
      lines.map {|line| line.slice(num_leading_spaces)}
    end

    def num_leading_spaces_to_remove; end


    def num_leading_spaces(line); end
  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```


```
class Listing
  attr_reader :source, :subsetter, :left_just
  def initialize(source:, subsetter:, left_justify: false)
    @source      = source
    @subsetter    = subsetter
    @left_just    = left_justify
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```



```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```



```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end
  def lines
    subset = subsetter.lines(source.lines)

    if left_just
      return justify(subset)
    end
    subset
  end
end
```


```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end

  def lines
    subset = subsetter.lines(source.lines)

    justifier.justify(subset)
  end
end
```




```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end
  def lines
    subset = subsetter.lines(source.lines)
    justifier.justify(subset)
  end
end
```



```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end
  def lines
    subset = subsetter.lines(source.lines)

    justifier.justify(subsetter.lines(source.lines))

  end
end
```



```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end
  def lines

    justifier.justify(subsetter.lines(source.lines))

  end
end
```

```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end
  def lines
    justifier.justify(subsetter.lines(source.lines))
  end
end
```

But, conditionals?

```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source      = source
    @subsetter    = subsetter
    @justifier    = justifier
  end
end
```

3 Code Line *or* Comment


```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source = source
    @subsetter = subsetter
    @justifier = justifier
  end
end
```

```
module Subset
```

```
  class LineNumber
```

```
    def lines(possibilities)
```

```
      specs = line_numbers.gsub(/['|']/, " ")
```

```
      specs.collect {|spec|
```

```
        if spec.include?('#')
```

```
          num_spaces = spec.delete("#").to_i
```

```
          (" " * num_spaces) + "# ..."
```

```
        else
```

```
          edges = spec.split('-').collect {|edge|
```

```
            range_of_line_numbers = (edges.first..edges.last)
```

```
            range_of_line_numbers.collect {|line_number|
```

```
              end
```

```
            }.flatten.compact
```

```
          end
```

Code Line or Comment

```
class Listing
```

```
  attr_reader :source, :subsetter, :justifier
```

```
  def initialize(source:, subsetter:, justifier:)
```

```
    @source = source
```

```
    @subsetter = subsetter
```

```
    @justifier = justifier
```

```
  end
```

```
end
```

```
end
```

```
end
```

3 Clump

```
module Subset
```

```
  class LineNumber
```

```
    def lines(possibilities)
```

```
      specs = line_numbers.gsub(/['|']/, "
```

```
      specs.collect {|spec|
```

```
        if spec.include?('#')
```

```
          num_spaces = spec.delete("#").to
```

```
          (" " * num_spaces) + "# ..."
```

```
        else
```

```
          edges = spec.split('-').collect(
```

```
            range_of_line_numbers = (edges.m
```

```
            range_of_line_numbers.collect {|
```

```
        end
```

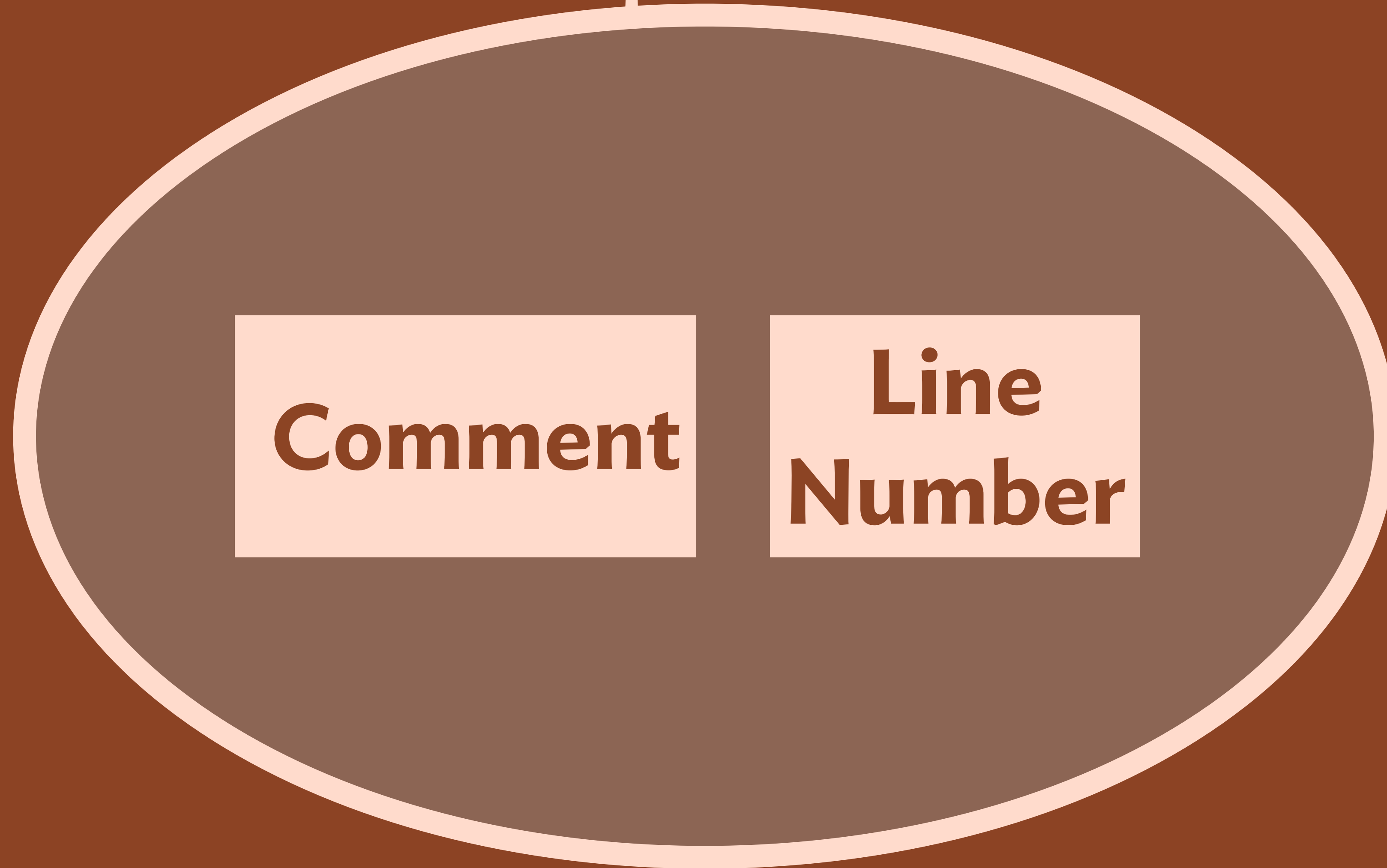
```
      }.flatten.compact
```

```
    end
```

Clump

Clump #lines

Clump #lines



C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.compact
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```


C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.compact
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```

C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.compact
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```

C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.c
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```

C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.c
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```

C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.c
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```

C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.compact
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```


C L U M P

```
class Clump
  # initialize with spec and input
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end

  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.compact
    end

    def expand_clump(spec)
      # ...
    end
  end
end
```

SUBSET

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          num_spaces = spec.delete("#").to_i
          (" " * num_spaces) + "# ..."
        else
          edges = spec.split('-').collect(&:to_i)
          range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
          range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
        end
      }.flatten.compact
    end
  end
end
```

SUBSET

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          num_spaces = spec.delete("#").to_i
          (" " * num_spaces) + "# ..."
        else
          edges = spec.split('-').collect(&:to_i)
          range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
          range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
        end
      }.flatten.compact
    end
  end
end
```

SUBSET

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')

        else

        end
      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')

          else

        end
      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          end
      }.flatten.compact
    end
  end
end
```



```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```



Polymorphic

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```

Role-playing

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#') ← Rule ??
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```

Tightly-coupled

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```



```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(
        specs.collect {|spec|
          if spec.include?('#')
            Clump::Comment.new(...)
          else
            Clump::LineNumber.new(...)
          end
        }.flatten.compact
      )
    end
  end
end
```



Factory!

CLUMP

```
class Clump
  def self.for(spec:, possibilities: [])
    if spec.include?('#')
      Clump::Comment
    else
      Clump::LineNumber
    end.new(spec: spec, input: possibilities)
  end
  # ...
end
```

CLUMP

```
class Clump
  def self.for(spec:, possibilities: [])
    if spec.include?('#')
      Clump::Comment
    else
      Clump::LineNumber
    end.new(spec: spec, input: possibilities)
  end
  # ...
end
```

C L U M P

```
class Clump
```

```
  def self.for(spec:, possibilities: [])  
    if spec.include?('#')  
      Clump::Comment  
    else  
      Clump::LineNumber  
    end.new(spec: spec, input: possibilities)  
  end  
  # ...
```

```
end
```


CLUMP

```
class Clump
  def self.lines(spec:, possibilities: [])
    self.for(spec: spec, possibilities: possibilities).lines
  end

  def self.for(spec:, possibilities: [])
    if spec.include?('#')
      Clump::Comment
    else
      Clump::LineNumber
    end.new(spec: spec, input: possibilities)
  end
  # ...
end
```

```
class Clump
  def self.lines(spec:, possibilities: [])
    self.for(spec: spec, possibilities: possibilities).lines
  end

  def self.for(spec:, possibilities: [])
    if spec.include?('#')
      Clump::Comment
    else
      Clump::LineNumber
    end.new(spec: spec, input: possibilities)
  end
  # ...
end
```

Factory-created

SUBSET

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|
        if spec.include?('#')
          Clump::Comment.new(...).lines
        else
          Clump::LineNumber.new(...).lines
        end
      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
      specs.collect {|spec|

      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ' ').split(",")
      specs.collect {|spec|
        Clump.lines(spec: spec, possibilities: possibilities)
      }.flatten.compact
    end
  end
end
```

```
module Subset
  class LineNumber
    def lines(possibilities)
      specs = line_numbers.gsub(/['|']/, "").gsub(/ /, ',').split(",")
      specs.collect {|spec|
        Clump.lines(spec: spec, possibilities: possibilities)
      }.flatten.compact
    end
  end
end
```

Reprise

1

```
class Listing
  attr_reader :filename, :repository, :tag, :git_cmd

  def initialize(filename:, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    if git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
      git_cmd.show.split("\n")
    else
      File.read(filename).split("\n")
    end
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

2

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      edges = spec.split('-').collect(&:to_i)
      individual_numbers = (edges.min.to_i..edges.max.to_i).to_a
      individual_numbers.collect {|i| all_lines[i - 1]}.compact
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

3

```
class Listing
  attr_reader :filename, :line_numbers, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    if line_numbers
      return lines_to_print(all_lines)
    end
    all_lines
  end

  private
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
```

4

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd

  def initialize(filename:, line_numbers: nil, left_just: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_just
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end

  private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end

  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end

  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end

  def num_leading_spaces(line)
    line[/\A */].size
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    git #{git_dir} show #{tagname}:#{filename}`
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end

# plus 90+ lines of error handling
```

Procedure

Lines: 83
Execution
Paths: 16

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag:
nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
  private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('##')
        num_spaces = spec.delete("##").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/\A */].size
  end
end
end
class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
end
# plus 90+ lines of error handling
```

Procedure

Lines: 83

Execution

Paths: 16

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag:
nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
  private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?(' #')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/\A */].size
  end
end
#####
class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ lines of error handling
```

OO

Procedure

Lines: 83
Execution
Paths: 16

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justif: false, repository: nil, tag:
nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justif
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('}')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/\A */].size
  end
end
class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ Lines of error handling
```

OO

Source:
File
GitTag

```
module Source
  class File
    attr_reader :filename

    def initialize(filename:)
      @filename = filename
    end

    def lines
      File.read(filename).split("\n")
    end
  end
  class GitTag
    def self.git_cmd
      GitCmd.new
    end

    attr_reader :filename, :tagname, :repository, :git_cmd

    def initialize(filename:, repository:, tag:, git_cmd: self.class.git_cmd)
      @git_cmd = git_cmd
      git_cmd.repository = repository
      git_cmd.tagname = tag
      git_cmd.filename = filename
    end
    def lines
      git_cmd.show.split("\n")
    end
  end
  class GitCmd
    attr_accessor :repository, :tagname, :filename

    def show
      `git #{git_dir} show #{tagname}:#{filename}`
    end

    def git_dir
      %Q[--git-dir=#{repository}]
    end
  end
end
```

Procedure

Lines: 83
Execution
Paths: 16

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag:
nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('##')
        num_spaces = spec.delete("##").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/\A */].size
  end
end
class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ lines of error handling
```

OO

Source:

Subset:

Everything
LineNumber

```
module Source
  class File
    attr_reader :filename

    def initialize(filename:)
      @filename = filename
    end

    def lines
      File.read(filename).split("\n")
    end
  end
end
class GitTag
  def set
    GitCmd
  end
end
module Subset
  class Everything
    def lines(everything)
      everything
    end
  end
  class LineNumber
    attr_reader :line_numbers
    def initialize(line_numbers:)
      @line_numbers = line_numbers
    end

    def lines(possibilities)
      clump_specs.collect {|spec| clump_for(spec, possibilities) }.flatten.compact
    end

    def clump_specs
      line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    end

    def clump_for(spec, possibilities)
      Clump.lines(spec: spec, possibilities: possibilities)
    end
  end
end
end
```


Procedure

Lines: 83
Execution
Paths: 16

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/\A */].size
  end
end
class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ lines of error handling
```

OO

Source:

Subset:

Clump:

Comment

LineNumber

```
module Source
  class File
    attr_reader :filename

    def initialize(filename:)
      @filename = filename
    end

    def lines
      File.read(filename).split("\n")
    end
  end
end
class GitTag
  def self.new(filename, tagname)
    GitCmd.new(filename, tagname).show
  end
end
module Subset
  class Everything
    def lines(everything)
      everything
    end
  end
  class LineNumber
    attr_reader :line_numbers
    def initialize(line_numbers:)
      @line_numbers = line_numbers
    end
    def lines(possibilities)
      clump = possibilities.flat_map {|p| p.lines }
      clump
    end
  end
  class Comment
    attr_reader :spec, :input
    def initialize(spec:, input: [])
      @spec = spec
      @input = input
    end
    def lines
      if spec.include?('#')
        Clump::Comment
      else
        Clump::LineNumber
      end
    end
  end
end
class Clump
  def self.lines(spec:, possibilities: [])
    self.for(spec: spec, possibilities: possibilities).lines
  end
  def clump(spec:, possibilities: [])
    if spec.include?('#')
      Clump::Comment
    else
      Clump::LineNumber
    end
  end
  attr_reader :spec, :input
  def initialize(spec:, input: [])
    @spec = spec
    @input = input
  end
  class LineNumber < Clump
    def lines
      expand_clump(spec).collect {|i| input[i - 1]}.compact
    end
  end
  def expand_clump(spec)
    edges = spec.split('-').collect(&:to_i)
    (edges.min.to_i..edges.max.to_i).to_a
  end
  class Comment < Clump
    def lines
      num_spaces = spec.delete("#").to_i
      (" " * num_spaces) + "# ..."
    end
  end
end
```


Procedure

Lines: 83
Execution
Paths: 16

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/\A */].size
  end
end
class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ lines of error handling
```

OO

Source:

Subset:

Clump:

Justification:

None

BlockLeft

Procedure

Lines: 83
Execution
Paths: **16**

```

class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end

  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end

  private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end

  def file_lines
    File.read(filename).split("\n")
  end

  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('|')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end

  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end

  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end

  def num_leading_spaces(line)
    line[/^\A */].size
  end
end

class GitCmd
  attr_accessor :repository, :tagname, :filename

  def show
    git #{git_dir} show #{tagname}:#{filename}
  end

  private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ lines of error handling

```

00

Source:

Subset:

Clump:

Justification:

class Listing

```
attr_reader :source, :subsetter, :justifier
def initialize(source:, subsetter:, justifier:)
  @source = source
  @subsetter = subsetter
  @justifier = justifier
end
def lines
  justifier.justify(subsetter.lines(source.lines))
end
end
```

Procedure

Lines: 83
Execution
Paths: 16

```
class Listing
  attr_reader :filename, :line_numbers, :left_just, :repository, :tag, :git_cmd
  def initialize(filename:, line_numbers: nil, left_justify: false, repository: nil, tag: nil, git_cmd: nil)
    @filename = filename
    @line_numbers = line_numbers
    @left_just = left_justify
    @repository = repository
    @tag = tag
    @git_cmd = git_cmd
  end
  def lines
    all_lines =
      if git_cmd
        git_lines
      else
        file_lines
      end
    subset =
      if line_numbers
        lines_to_print(all_lines)
      else
        all_lines
      end
    if left_just
      return justify(subset)
    end
    subset
  end
private
  #####
  Reading
  #####
  def git_lines
    git_cmd.repository = repository
    git_cmd.tagname = tag
    git_cmd.filename = filename
    git_cmd.show.split("\n")
  end
  def file_lines
    File.read(filename).split("\n")
  end
  #####
  Subsetting
  #####
  def lines_to_print(all_lines)
    specs = line_numbers.gsub(/['|']/, "").gsub(/ /, '').split(",")
    specs.collect {|spec|
      if spec.include?('#')
        num_spaces = spec.delete("#").to_i
        (" " * num_spaces) + "# ..."
      else
        edges = spec.split('-').collect(&:to_i)
        range_of_line_numbers = (edges.min.to_i..edges.max.to_i).to_a
        range_of_line_numbers.collect {|i| all_lines[i - 1]}.compact
      end
    }.flatten.compact
  end
  #####
  Justification
  #####
  def justify(lines)
    lines.map {|line| line.slice(num_leading_spaces_to_remove(lines)..-1) || "" }
  end
  def num_leading_spaces_to_remove(lines)
    @num ||=
      lines.reduce(999999) {|current_min, line|
        line.empty? ? current_min : [current_min, num_leading_spaces(line)].min
      }
  end
  def num_leading_spaces(line)
    line[/\A */].size
  end
end
class GitCmd
  attr_accessor :repository, :tagname, :filename
  def show
    `git #{git_dir} show #{tagname}:#{filename}`
  end
private
  def git_dir
    %Q[--git-dir=#{repository}]
  end
end
# plus 90+ lines of error handling
```

OO

Source:

Subset:

Clump:

Justification:

class Listing



Factories

Procedure

Lines: 83
Execution
Paths: **16**

OO

Source:

Subset:

Clump:

Justification:

class Listing



Procedure

Lines: 83
Execution
Paths: **16**

OO

Source:

Subset:

Clump:

Justification:

class Listing



Procedure

Lines: 83
Execution
Paths: **16**

OO

Total Lines: 134

Source:

Subset:

Clump:

Justification:

class Listing

```
module Source
  class File
    attr_reader :filename

    def initialize(filename:)
      @filename = filename
    end

    def lines
      File.open(filename).split("\n")
    end
  end

  class GitTag
    def self.all
      Git::Tags.all
    end
  end

  module Subset
    class Everything
      def lines(everything)
        everything
      end
    end

    class LineNumber
      attr_reader :line_numbers
      def initialize(line_numbers)
        @line_numbers = line_numbers
      end

      def lines(possibilities)
        clump_specs.collect {|spec| clump_for(spec, possibilities) }.flatten.compact
      end

      def clump_for(spec, possibilities)
        self.for(spec: spec, possibilities: possibilities).lines
      end

      def self.for(spec, possibilities)
        if spec.include? '#'
          Clump::Comment
        else
          Clump::LineNumber
        end.new(spec: spec, input: possibilities)
      end
    end

    attr_reader :spec, :input
    def initialize
      @spec =
      @input =
    end

    class LineNumber
      def lines
        expand_lines
      end

      def expand_lines
        edges =
        (edges,
        end
      end

      class Comment
        def lines
          num_spaces
          (" " *
          end
        end
      end
    end
  end

  module Justification
    class None
      def self.justify(lines)
        lines
      end
    end

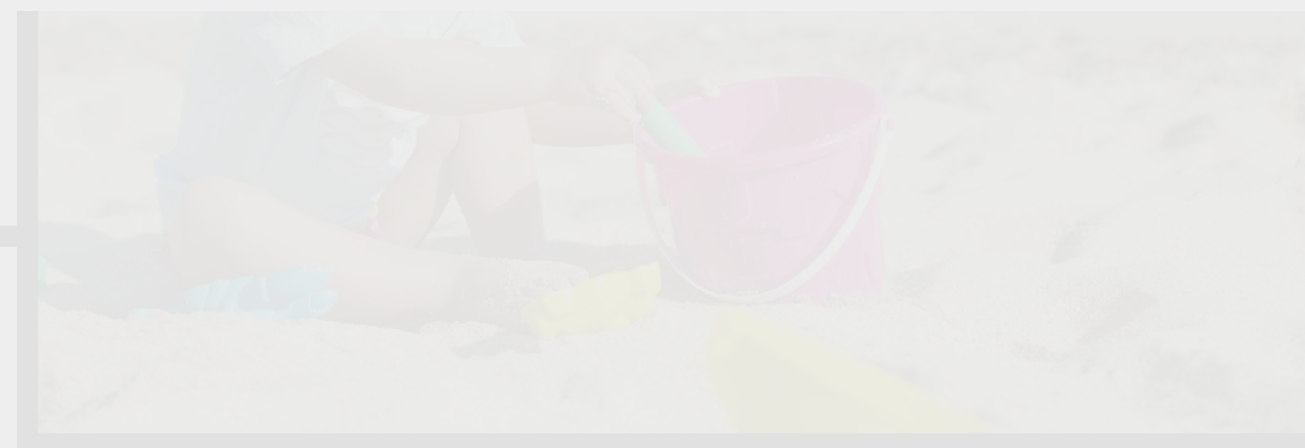
    class BackLeft
      def self.justify(lines)
        new_lines = still
      end
    end

    attr_reader :lines
    def initialize(lines)
      @lines = lines
    end

    def justify
      lines
    end

    private
    def num_spaces
      @num_spaces ||=
      line_numbers
    end
  end

  def num_spaces
    line_numbers
  end
end
```



Procedure

Lines: 83
Execution
Paths: **16**

OO

Total Lines: 134
Execution Paths: **1**

Source:
Subset:
Clump:

Justification:

class Listing

```
attr_reader :source, :subsetter, :justifier
def initialize(source:, subsetter:, justifier:)
  @source = source
  @subsetter = subsetter
  @justifier = justifier
end
def lines
  justifier.justify(subsetter.lines(source.lines))
end
end
```

Procedure

Lines: 83
Execution
Paths: **16**

OO

Source:

Subset:

Clump:

Justification:

class Listing

Total Lines: 134
Execution Paths: **1**

```
module Source
  class File
    attr_reader :filename

    def initialize(filename:)
      @filename = filename
    end

    def lines
      File.open(filename).split("\n")
    end
  end

  class GitTag
    def self.all
      Git::Tags.all
    end
  end

  class Subset
    class Everything
      def lines(everything)
        everything
      end
    end

    class LineNumber
      attr_reader :line_numbers

      def initialize(line_numbers)
        @line_numbers = line_numbers
      end

      def lines(possibilities)
        clump = possibilities.select { |line| line =~ /\d+/ }
        clump.map { |line| line.gsub(/\d+/, '') }
      end
    end

    class Clump
      def self.lines(spec, possibilities: [])
        self.for(spec: spec, possibilities: possibilities).lines
      end

      def self.for(spec: spec, possibilities: possibilities)
        if spec.include? '#'
          Clump::Comment.new(spec).lines(possibilities)
        else
          Clump::LineNumber.new(spec).lines(possibilities)
        end
      end
    end

    attr_reader :spec, :input
    def initialize(spec:, input: [])
      @spec = spec
      @input = input
    end

    class LineNumber
      def lines
        expand_lines
      end
    end

    def expand_lines
      edges =
        (edges =
          end
        end
      end

    class Comment
      def lines
        num_spaces =
          (" " * num_spaces)
        end
      end
    end
  end
end

module Justification
  class None
    def self.justify(lines)
      lines
    end
  end

  class BackLeft
    def self.justify(lines)
      new_lines =
        new_lines
      end
    end

    attr_reader :lines
    def initialize(lines)
      @lines = lines
    end

    def justify
      lines
    end

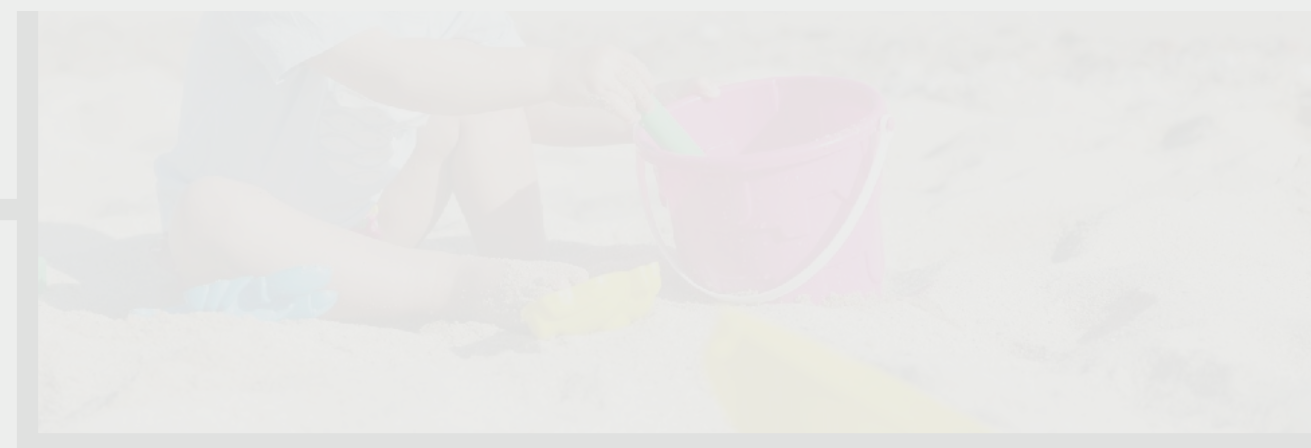
    private
    def num_spaces
      @num_spaces ||=
        line[0].length
      end
    end
  end

  def num_spaces
    line[0].length
  end
end

class Listing
  attr_reader :source, :subsetter, :justifier

  def initialize(source:, subsetter:, justifier:)
    @source = source
    @subsetter = subsetter
    @justifier = justifier
  end

  def lines
    justifier.justify(subsetter.lines(source.lines))
  end
end
```



Procedure

Lines: 83
Execution
Paths: **16**

OO

Source:

Subset:

Clump:

Justif

class Listing

Total Lines: 134
Execution Paths: **1**
Classes: 9

```
module Source
  class File
    attr_reader :filename

    def initialize(filename:)
      @filename = filename
    end

    def lines
      File.open(filename).split("\n")
    end
  end
end

class GitTag
  def self.all
    Git::Tags.all
  end
end

module Subset
  class Everything
    def lines(everything)
      everything
    end
  end

  class LineNumber
    attr_reader :line_number

    def initialize(line_numbers)
      @line_numbers = line_numbers
    end

    def lines(possibilities)
      possibilities
    end
  end

  class Clump
    def self.lines(spec, possibilities: [])
      self.for(spec: spec, possibilities: possibilities).lines
    end

    def self.for(spec: spec, possibilities: possibilities)
      if spec.include? '#'
        Clump::Comment
      else
        Clump::LineNumber
      end
    end
  end

  attr_reader :spec, :input
  def initialize(spec:, input: [])
    @spec = spec
    @input = input
  end

  class LineNumber
    def lines
      expand
    end

    def expand
      edges =
        (edges)
      end
    end

    class Comment
      def lines
        num_spaces
      end
    end
  end
end

module Justification
  class None
    def self.justify(lines)
      lines
    end
  end

  class BackLeft
    def self.justify(lines)
      new_lines = still
    end
  end

  attr_reader :lines
  def initialize(lines)
    @lines = lines
  end

  def justify
    lines
  end

  private
  def num_spaces
    @num_spaces
  end
end

class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source = source
    @subsetter = subsetter
    @justifier = justifier
  end

  def lines
    justifier.justify(subsetter.lines(source.lines))
  end
end
```

Procedure

Lines: 83
Execution
Paths: **16**

OO

Source:

Subset:

Clump:

Justif

class Listing

Total Lines: 134
Execution Paths: **1**
Classes: 9
Biggest: 18 loc

```
attr_reader :source, :subsetter, :justifier
def initialize(source:, subsetter:, justifier:)
  @source = source
  @subsetter = subsetter
  @justifier = justifier
end
def lines
  justifier.justify(subsetter.lines(source.lines))
end
end
```


Procedure

Lines: 83
Execution
Paths: **16**

OO

Source:

Subset:

Clump:

Justification:

class Listing




```
class Clump
  def self.for(spec:, possibiliti
    if spec.include?('#')
      Clump::Comment
    else
      Clump::LineNumber
    end.new(spec: spec, input: po
  end
  # ...
end
```




```
class Clump
  def self.for(spec:, possibiliti
    if spec.include?('#')
      Clump::Comment
    else
      Clump::LineNumber
    end.new(spec: spec, input: po
  end
  # ...
end
```



Factories are shared
Factories are isolated
Factories are simple
Factories are ignorable

OO

Source:

```
module Source
  class File
    attr_reader :filename

    def initialize(filename:)
      @filename = filename
    end

    def lines
      File.readlines(filename).split("\n")
    end
  end

  class GitTag
    def self.new
      GitTag.new
    end
  end

  class Subset
    class Everything
      def lines(everything)
        everything
      end
    end

    class LineNumber
      attr_reader :line_number
      def initialize(line_number)
        @line_number = line_number
      end
    end

    def lines(possibilities)
      clump = Clump.new(possibilities)
      clump.for(spec: spec, possibilities: possibilities).lines
    end
  end

  class Clump
    def self.new(spec:, possibilities: [])
      self.for(spec: spec, possibilities: possibilities).lines
    end

    def self.for(spec:, possibilities: [])
      if spec.include? '#'
        Clump::Comment.new(possibilities)
      else
        Clump::LineNumber.new(possibilities)
      end
    end

    attr_reader :spec, :input
    def initialize(spec:, input: [])
      @spec = spec
      @input = input
    end
  end

  class LineNumber
    def lines
      expand_edges =
        (edges =
          end
        end
      end

    class Comment
      def lines
        num_spaces =
          (" " * num_spaces)
        end
      end
    end
  end
end
```

Subset:

```
module Subset
  class Everything
    def lines(everything)
      everything
    end
  end

  class LineNumber
    attr_reader :line_number
    def initialize(line_number)
      @line_number = line_number
    end
  end

  def lines(possibilities)
    clump = Clump.new(possibilities)
    clump.for(spec: spec, possibilities: possibilities).lines
  end
end
```

Clump:

```
class Clump
  def self.new(spec:, possibilities: [])
    self.for(spec: spec, possibilities: possibilities).lines
  end

  def self.for(spec:, possibilities: [])
    if spec.include? '#'
      Clump::Comment.new(possibilities)
    else
      Clump::LineNumber.new(possibilities)
    end
  end

  attr_reader :spec, :input
  def initialize(spec:, input: [])
    @spec = spec
    @input = input
  end
end

class LineNumber
  def lines
    expand_edges =
      (edges =
        end
      end
    end

    class Comment
      def lines
        num_spaces =
          (" " * num_spaces)
        end
      end
    end
  end
end
```

Justification:

```
module Justification
  class None
    def self.justify(lines)
      lines
    end
  end

  class BackLeft
    def self.justify(lines)
      new_lines =
        new_lines =
        end
      end

    attr_reader :lines
    def initialize(lines)
      @lines = lines
    end
  end

  def justify
    lines =
    end

  private
  def num_spaces
    @num_spaces =
      line_number
    end
  end
end

def num_spaces
  line_number
end
```



class Listing

```
class Listing
  attr_reader :source, :subsetter, :justifier
  def initialize(source:, subsetter:, justifier:)
    @source = source
    @subsetter = subsetter
    @justifier = justifier
  end

  def lines
    justifier.justify(subsetter.lines(source.lines))
  end
end
```

OO

class Listing

```
attr_reader :source, :subsetter, :justifier
def initialize(source:, subsetter:, justifier:)
  @source = source
  @subsetter = subsetter
  @justifier = justifier
end
def lines
  justifier.justify(subsetter.lines(source.lines))
end
end
```



```
class Listing
  attr_reader :source, :subsetter, :justifier

  def initialize(source:, subsetter:, justifier:)
    @source = source
    @subsetter = subsetter
    @justifier = justifier
  end

  def lines
    justifier.justify(subsetter.lines(source.lines))
  end
end
```



```
class Listing
  attr_reader :source, :subsetter, :justifier

  def initialize(source:, subsetter:, justifier:)
    @source = source
    @subsetter = subsetter
    @justifier = justifier
  end

  def lines
    justifier.justify(subsetter.lines(source.lines))
  end
end
```

Anthropomorphic

Polymorphic

Loosely-coupled

Role-playing

Factory-created

Message-sending

Thanks

Credits

door knobs and handles

<https://www.flickr.com/photos/nooner/2095466779/>

<https://www.flickr.com/photos/andybutkaj/1495839731/>

<https://www.flickr.com/photos/brenneman/5896515979/>

<https://www.flickr.com/photos/31050090@No3/2914850564/>

<https://www.flickr.com/photos/themacinator/3259887286/>

https://www.flickr.com/photos/stuart_spivack/932401578/

anthropomorphism <https://www.flickr.com/photos/clownhousethethird/2620514161/>

monarch <https://www.flickr.com/photos/pmillera4/16071290737/>

viceroy <https://www.flickr.com/photos/157813391@No5/40155797580/>

lab blue toy <https://www.flickr.com/photos/studiostein/3224148489/>

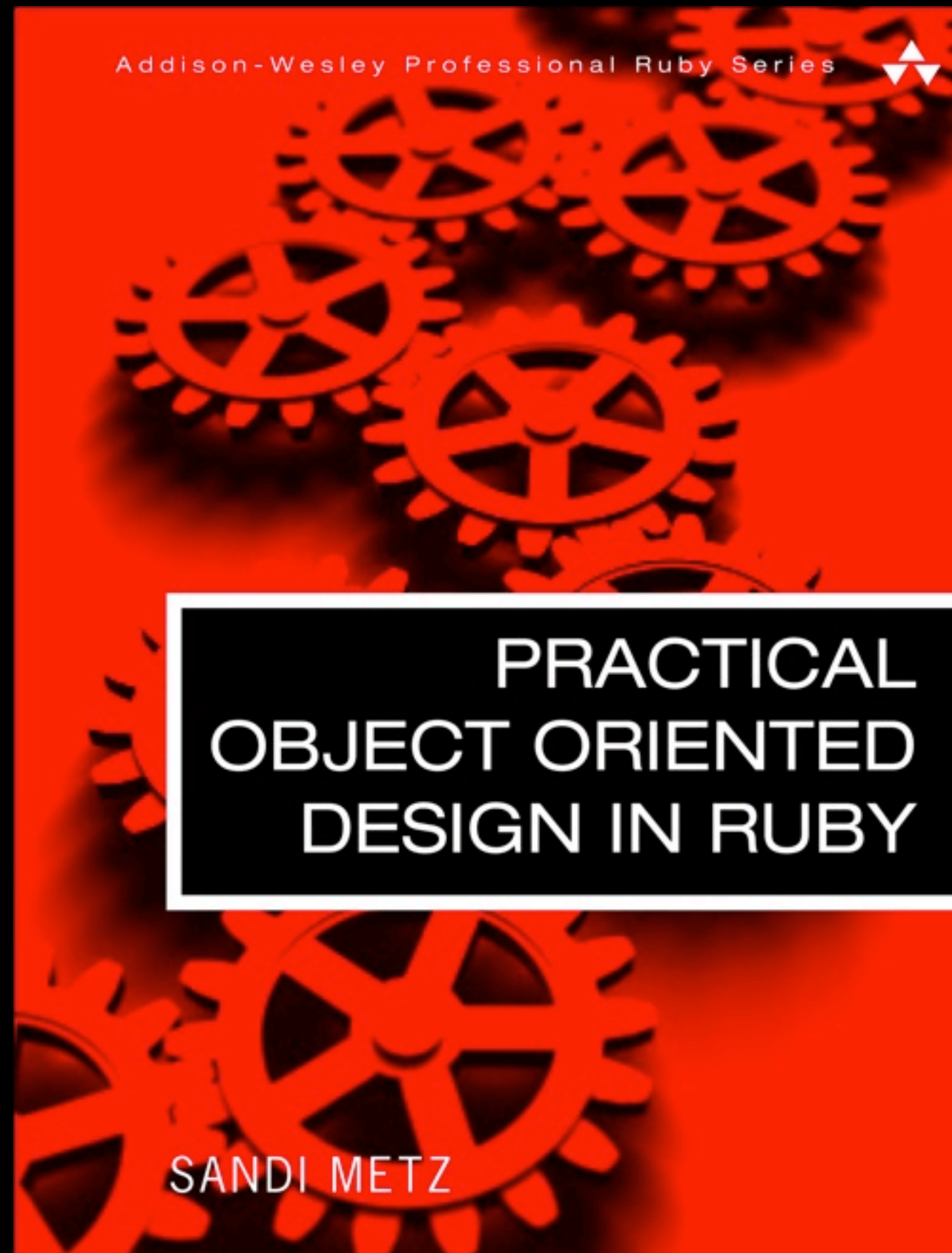
volleyball <https://www.flickr.com/photos/sangudo/24289743487/in/album-72157688506458842/>

big factory <https://www.flickr.com/photos/nobusue/2773870461/>

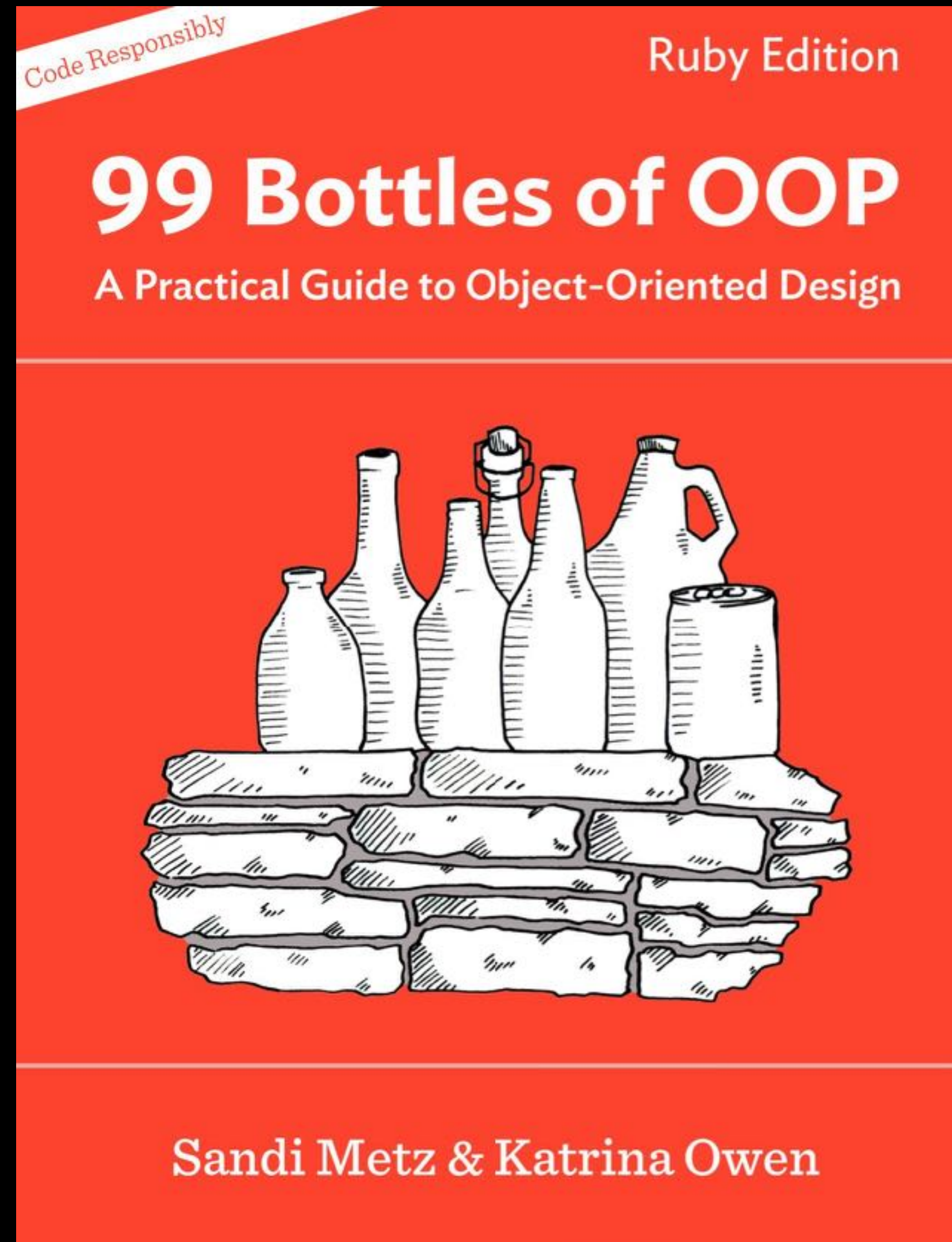
child on beach <https://www.flickr.com/photos/nicolasruh/26229103209/>

no evil <https://www.flickr.com/photos/pmillera4/14214119062/>

<http://poodr.com>



<http://99bottlesbook.com>





Sandi Metz

@sandimetz

<http://sandimetz.com>