Ruby, Rails & Go

Two (and a half) sides of the same coin

99designs®

whoisjohnbarton.com

github.com/joho

@johnbarton

2007

Ruby on Rails!

“the functionality of Rails came as extractions of a real application, not of a “what somebody might need some day” fantasy, so prevalent in framework design.

[...] And of course the joy of working with a technology so uniquely aligned with our thoughts on software development.”

–*David Heinemeier Hansson*

“This extraction-driven nature of Rails attracted a culture of practical programmers with a zeal for delivery”

–*David Heinemeier Hansson*

“Instead of emphasizing the what, I want to emphasize the how part: how we feel while programming. [...]

I didn't work hard to make Ruby perfect for everyone, because you feel differently from me. No language can be perfect for everyone”

–*Yukihiro Matsumoto*

“In our daily lives as programmers, we process text strings a lot. So I tried to work hard on text processing, namely the string class and regular expressions.”

–*Yukihiro Matsumoto*

Time Passes

2013

"Go's purpose is therefore *not* to do research into programming language design; it is to improve the working environment for its designers and their coworkers."

–Rob Pike (Inventor of Go)

“Some programmers find it fun to work in; others find it unimaginative, even boring.”

–Rob Pike (Inventor of Go)

```
GET /
Content-Type: application/json
200 OK
{"Hello": "World"}
```

```ruby
require 'rubygems'
require 'sinatra'
require 'json'

get '*' do
  content_type :json

  { "Hello" => "World" }.to_json
end
```

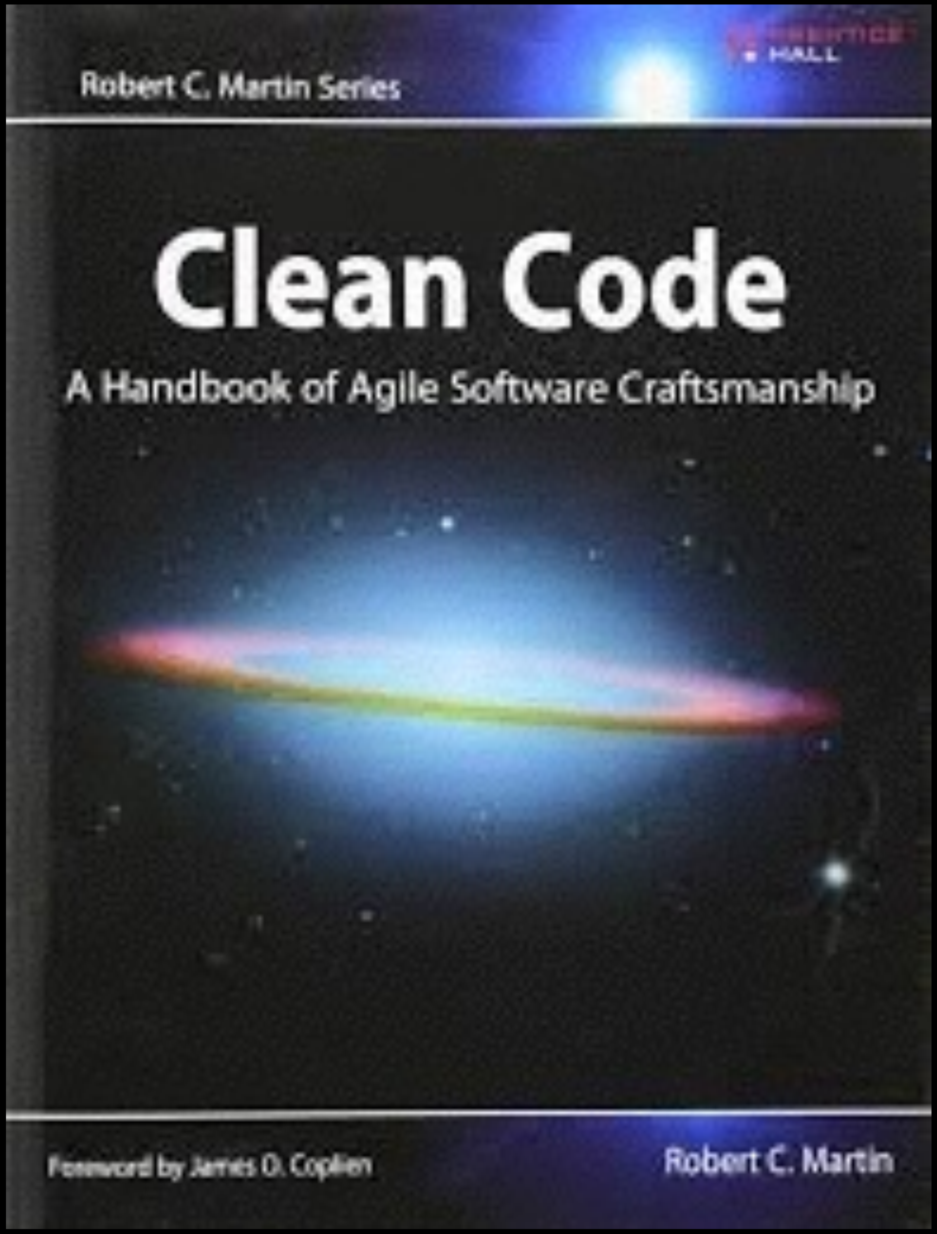
```go
package main

import (
  "encoding/json"
  "fmt"
  "net/http"
)

func main() {
  http.HandleFunc("/", func(w http.ResponseWriter,
    r *http.Request) {
    w.Header().Set("Content-Type", "application/json")

    json, _ := json.Marshal(map[string]string{
      "Hello": "World",
    })
    fmt.Fprint(w, string(json))
  })

  http.ListenAndServe(":4567", nil)
}
```

Duck Typing

```ruby
foo.bar if foo.respond_to?(:bar)
```

```go
type Barrable interface {
  Bar()
}
barrable, ok := foo.(Barrable)
if ok {
  barrable.Bar()
}
```

Error Handling



Robert C. Martin Series

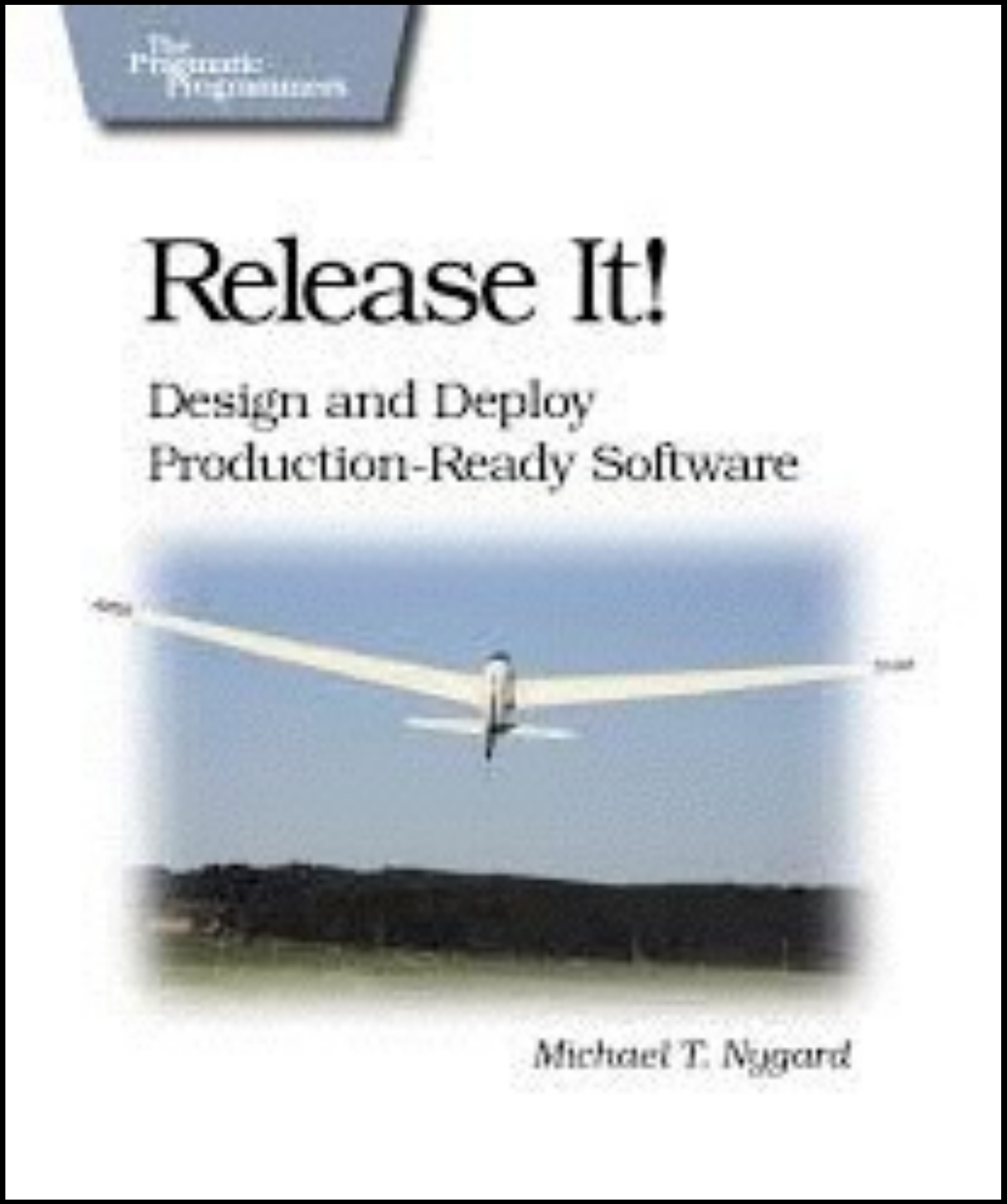
PRENTICE HALL

Clean Code

A Handbook of Agile Software Craftsmanship

Foreword by James O. Coplien

Robert C. Martin

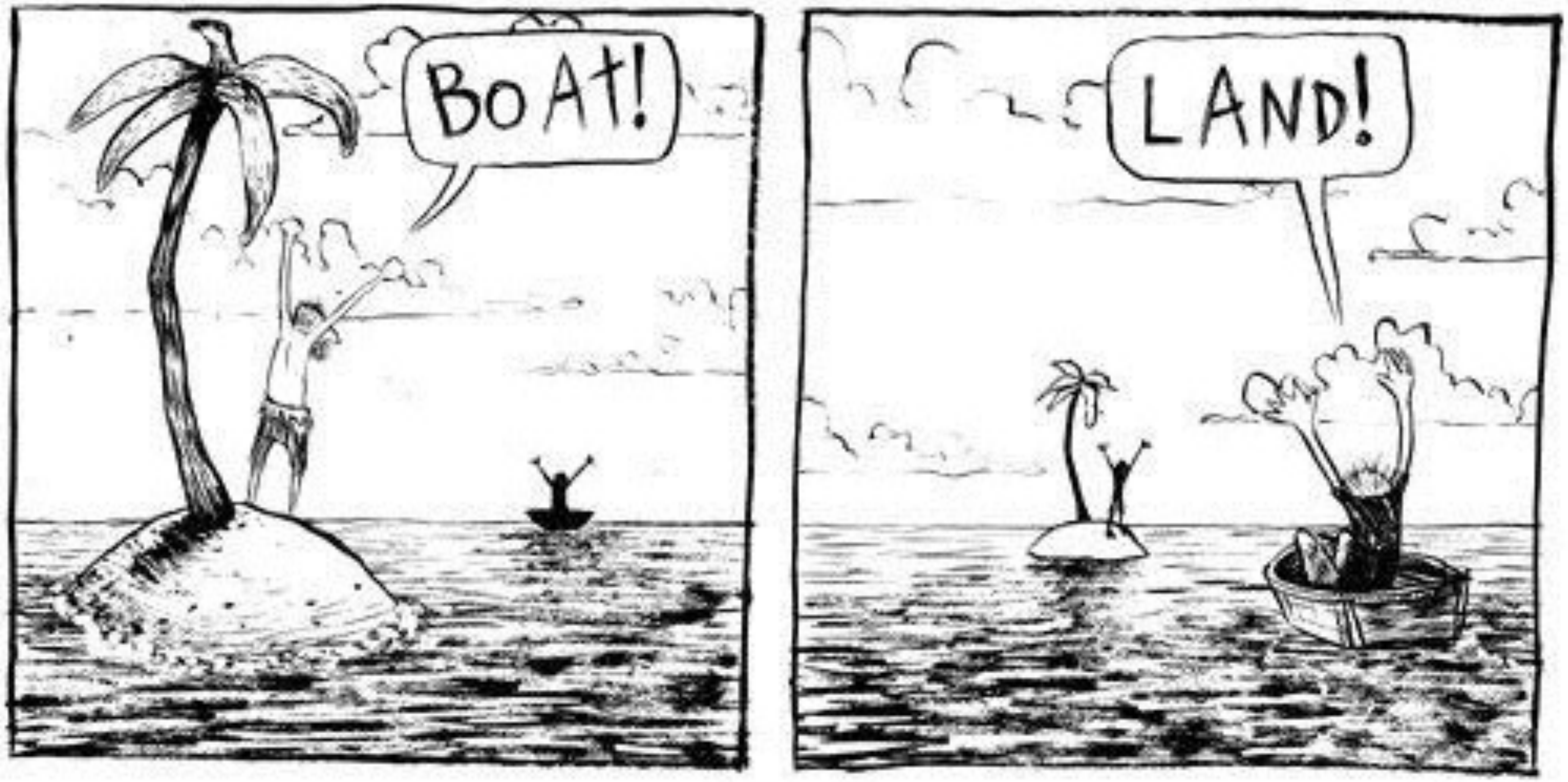


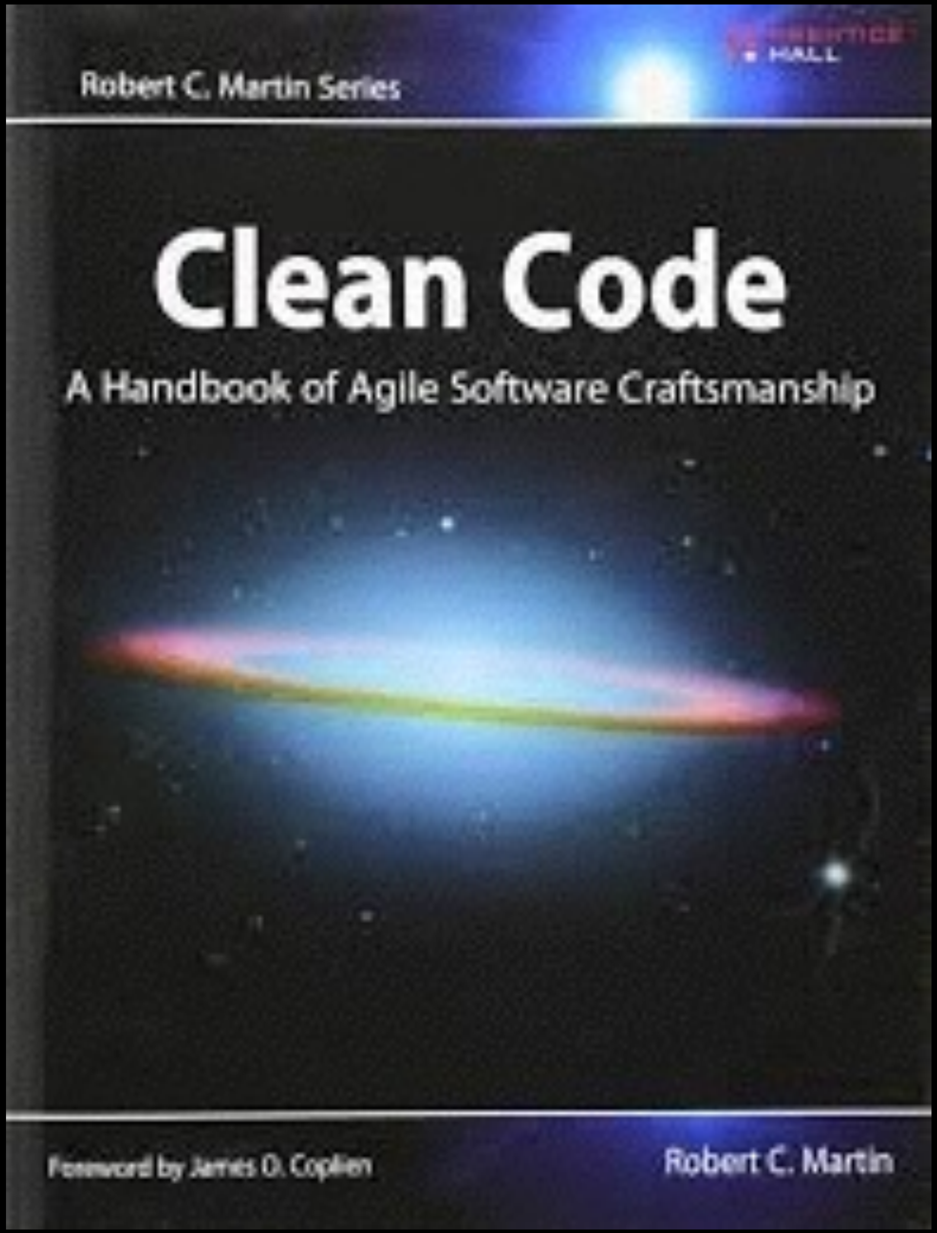
The Pragmatic Programmers

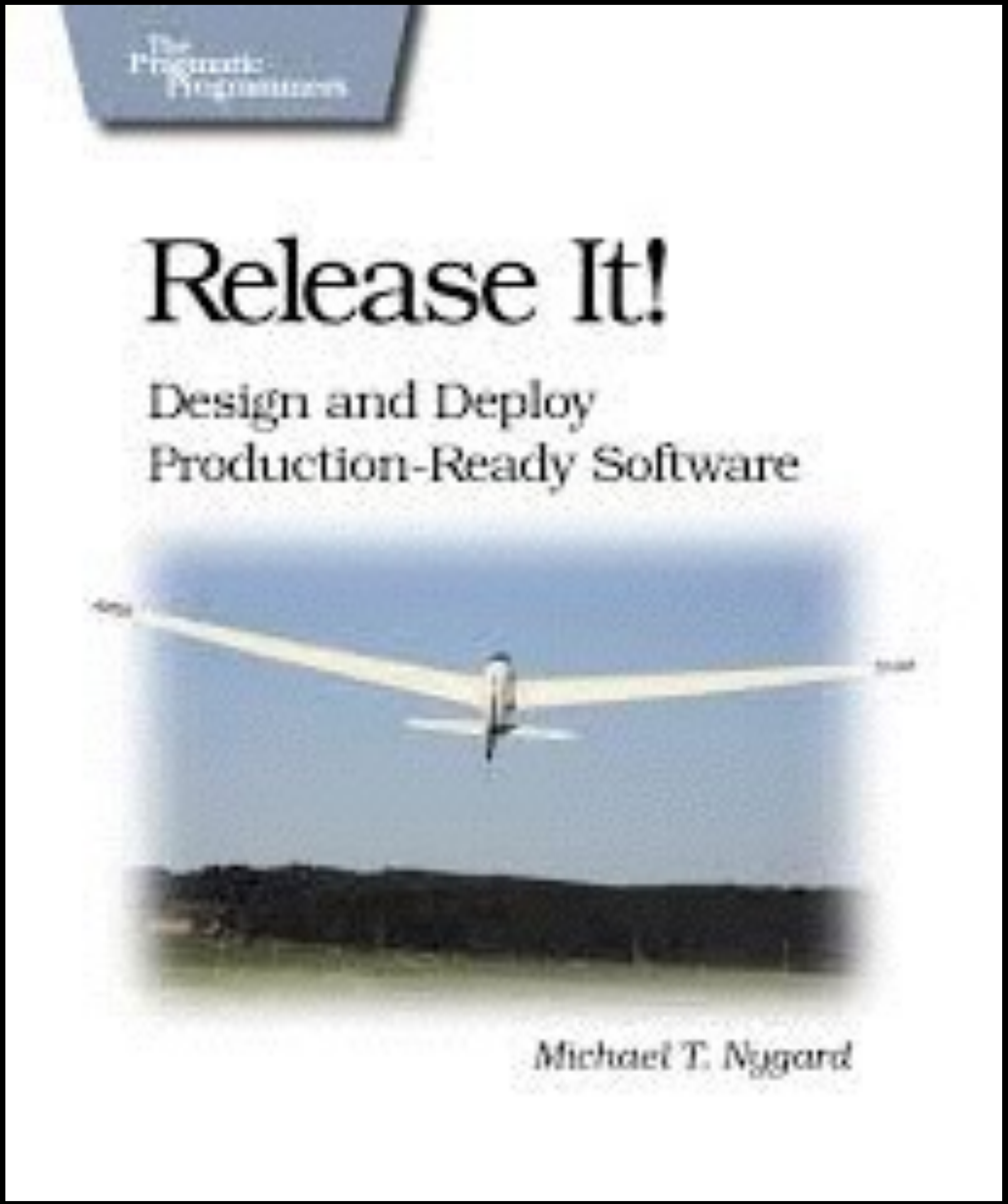
Release It!

Design and Deploy Production-Ready Software

Michael T. Nygard


BoAt!
LAND!


Robert C. Martin Series
Prentice Hall
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin


The Pragmatic Programmers
Release It!
Design and Deploy Production-Ready Software
Michael T. Nygard

```ruby
begin
  do_something_safe
  do_something_dangerous!
  keep_doing_safe_things
rescue => e
  # what now?
end
```

```go
package main

func main() {
  doSomethingSafe()
  err := doSomethingDangerous()
  if err != nil {
    // handle error
    return
  }
  keepDoingSafeThings()
}
```

Metaprogramming

Concurrency

POST /
uri encoded
Retry 5 times

```ruby
require 'net/http'

url = URI.parse('http://api.flakywebservice.com/')
http = Net::HTTP.new(url.host, url.port)
http.read_timeout = 600  # be very patient
res = nil

retries = 5
begin
  http.start{|conn|
    req = Net::HTTP::Post.new("foo" => "bah")
    req.set_form_data(params)
    res = conn.request(req)
  }
rescue Timeout::Error, Errno::EINVAL, Errno::ECONNRESET, EOFError,
       Net::HTTPBadResponse, Net::HTTPHeaderSyntaxError,
Net::ProtocolError
  sleep 3
  retry if (retries -= 1) > 0
end
# do something with response
```

```
func main() {
  client := &http.Client{}
  uri := "http://api.flakywebservice.com/"

  data := url.Values{"foo": {"bar"}}
  r, _ := http.NewRequest("POST", uri,
     bytes.NewBufferString(data.Encode()))

  attempts, maxAttempts := 0, 5
  var (
    resp *http.Response
    err  error
  )
  retries := 5
  for i := 0; i < retries; i++ {
    resp, err = client.Do(r)
    netErr, conversionOK := err.(net.Error)
    if err == nil || conversionOK && netErr.Temporary() {
      break
    }
    time.Sleep(3)
  }
  if err != nil {
    panic(err)
  }
   // do something with resp here
}
```

Community

Ye Olde Ruby

- Shitty package management (vendor everything)
- A million acts_as_taggable, tags_on_steroids, etc rails plugins
- Lots of frontend centric devs moving more backend
- Great if you want to ship fast and buy RAM and wake up at night

Current Go

- Shitty package management (vendor everything)
- A million exponential backoff packages
- Lots of ops people moving towards app development
- Great if you like sleep and cheap hosting and aren't in such a rush

JB's Rules of Thumb

- Are people and their "requirements" going to ruin your day? Use Ruby
- Are computers and their "flakiness" going to ruin your day? Use Go

Golang Melbourne

- http://www.meetup.com/golang-mel/
- https://twitter.com/golangmel
- Next meetup: 2nd September @ 99designs