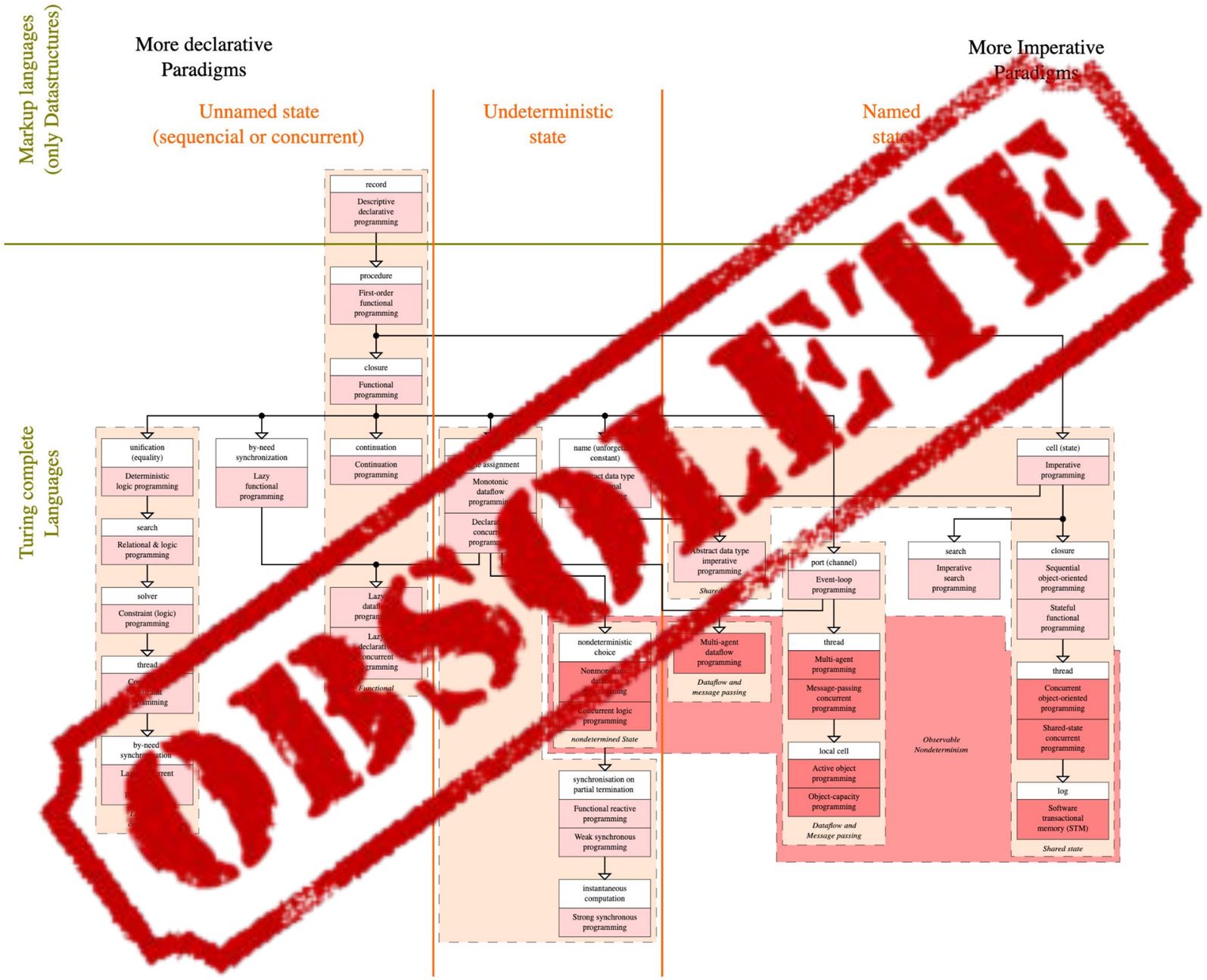
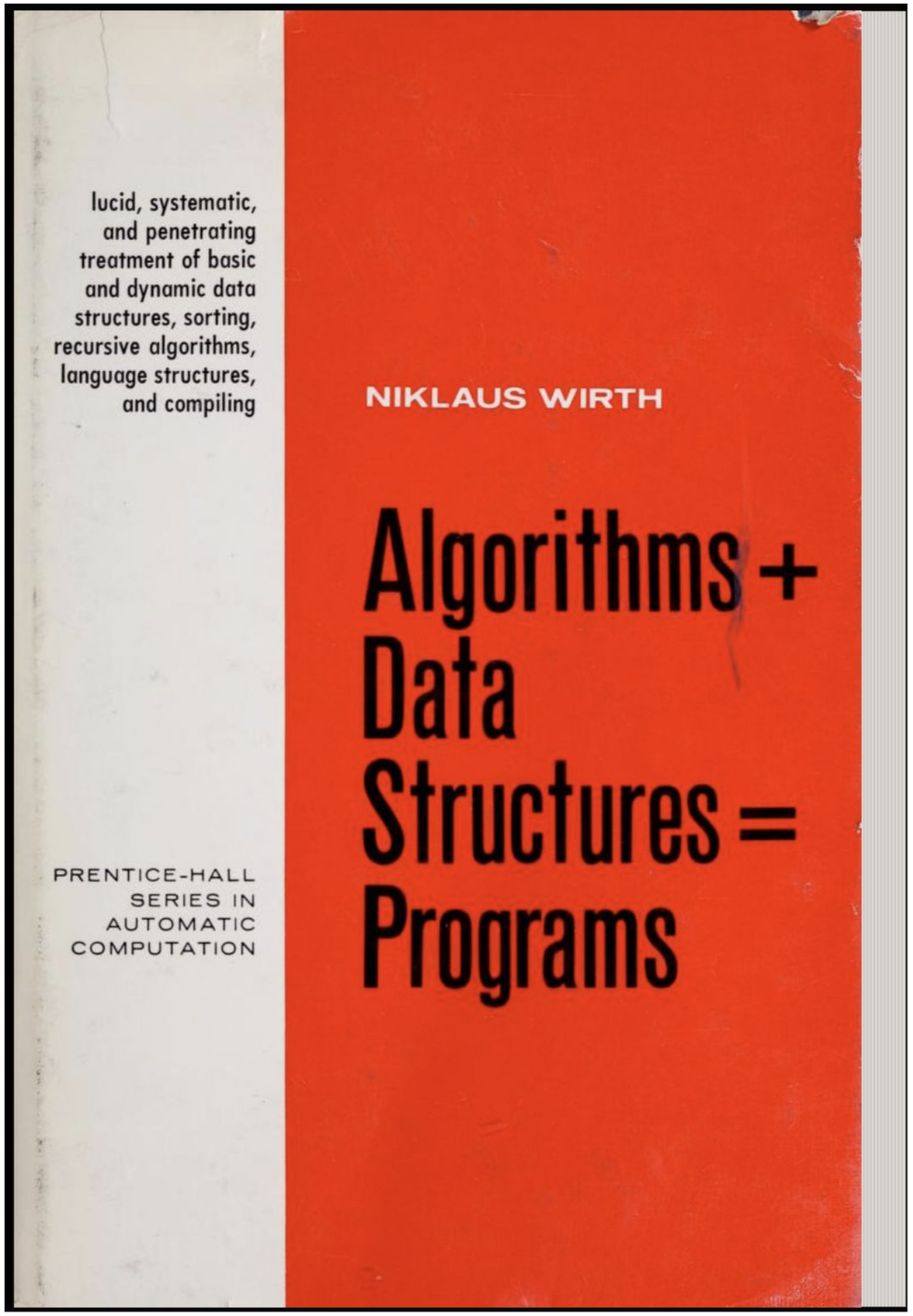

CHAMPIONS OF CUSTOMER SERVICE
zendesk

Programming Paradigms






```
{ key: value }
```

```
{ "key" => "value" }
```

Hash-Oriented Programming!

Hash-Oriented Programming

Programs are treated as a sequence of

Hash transformations

```ruby
class ArticlesController < ApplicationController
  def index
    @articles = Article.all
  end

  def show
    @article = Article.find(params[:id])
  end

  def new
    @article = Article.new
  end

  def create
    @article = Article.new(article_params)

    if @article.save
      redirect_to @article
    else
      render :new, status: :unprocessable_entity
    end
  end

  private
    def article_para
      params.require(:article).permit(:title, :body)
    end
end
```

Copy

Hash

Hash

Hash

Hi, I’m Łukasz 👋

Joined Base (now Zendesk Sell) in December 2013

Started as… Windows Phone Developer

Two months later moved to backend in… C#

In 2016 started serious Ruby & Rails development

In 2017-2019 coded both in Java and in R&R

In 2019 - 2020 worked on my most important Ruby project to date - “Sell SKU”

Since Q2 2020 - Team Lead @ Zendesk Sell

zendesk

Controller Action

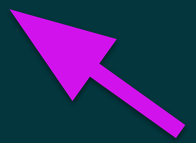
```ruby
def create
 action_options = {
   query_sanitizer: InputSanitizer::V2::QuerySanitizer,
   payload_sanitizer: Visit::CreateSanitizer
 }

 chassis_action(action_options) do |request, response|
   begin
     visit = VisitBuilder.create(request.user, request.params[:data])
     response.item_created(Visit::V1Representer.new(visit))
   rescue ActiveRecord::RecordInvalid => e
     response.model_errors(e.record)
   end
 end
end
```

Hash

```java
@PostMapping(produces = APPLICATION_JSON)
public Chassis2Entity<ProductGoal> create(
        @RequestBody Chassis2Entity<ProductGoalDto> productGoal,
        Principal principal) {
    ProductGoal createdProductGoal = productGoalsService.createProductGoal(
        getUser(principal),
        productGoal.getData()
    );
    return new Chassis2Entity<>(createdProductGoal, RESOURCE_TYPE);
}
```

Input Validation

```ruby
class Visit::CreateSanitizer < InputSanitizer::V2::PayloadSanitizer
 datetime :visited_at
 string :resource_type, allow: [FS::LEAD, FS::CONTACT, FS::SALES_ACCOUNT]
 integer :resource_id, minimum: 1
 integer :outcome_id, minimum: 1
 custom :rep_latitude, converter: Visit::GeoCoordinateSanitizer.new
 custom :rep_longitude, converter: Visit::GeoCoordinateSanitizer.new
 string :resource_address, minimum: 1, maximum: 2048
 string :rep_location_verification_status, allow:
   RepLocationVerificationStatus::VALUES
 string :summary, minimum: 1, maximum: 65535
 string :rep_address, minimum: 1, maximum: 2048
end

…

# Controller code
cleaned_params = Visit::CreateSanitizer.new(params).cleaned
```



Hash

```java
public interface ProductGoalCreateParams {
   @NotNull
   @Min(1L)
   Long getAssigneeId();

   @NotNull
   @Min(1L)
   Long getProductId();

   @NotNull
   @Min(1L)
   Long getProviderId();

   @NotNull
   @DecimalMin(value = "0.0", inclusive = false)
   @DecimalMaxScale(2)
   BigDecimal getQuota();

   @NotNull
   LocalDate getStartDate();

   @NotNull
   LocalDate getEndDate();
}

…

// ServiceObject code
var violations = validator.validate(
    createParams,
    ProductGoalCreateParams.class);
if (!violations.isEmpty()) {
   throw new ConstraintViolationException(violations);
}
```

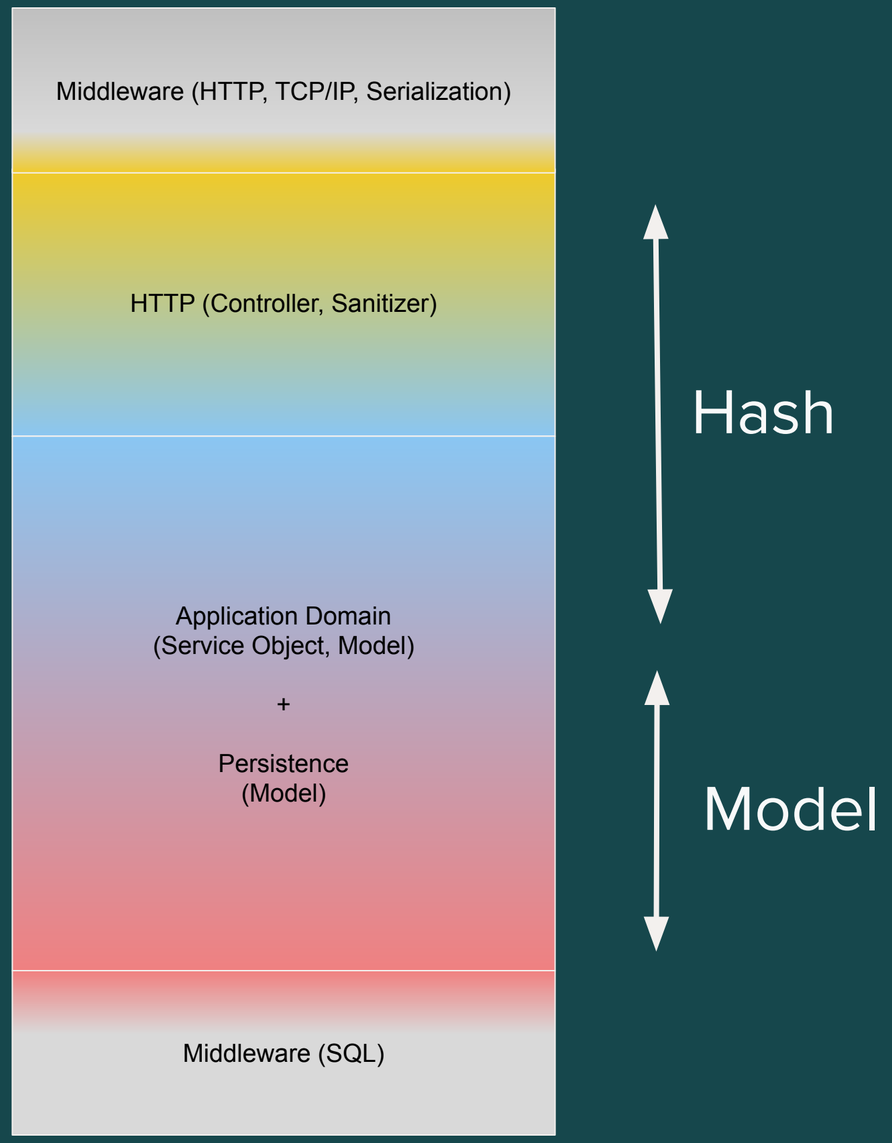
Persistence

```ruby
class VisitBuilder < Struct.new(:user, :params)
 attr_reader :visit

 def create
   build_visit
   validate
   in_transaction do
     save
   end
   visit
 end

 def build_visit
   @visit = Visit.new(params)
   @visit.creator_id = user.id
   @visit.permissions_holder_id = user.id
   @visit.account_id = user.account_id
 end

 def validate
   visit.valid?
   raise ActiveRecord::RecordInvalid.new(visit) unless visit.errors.empty?
 end

 def save
   visit.save!
 end
end
```

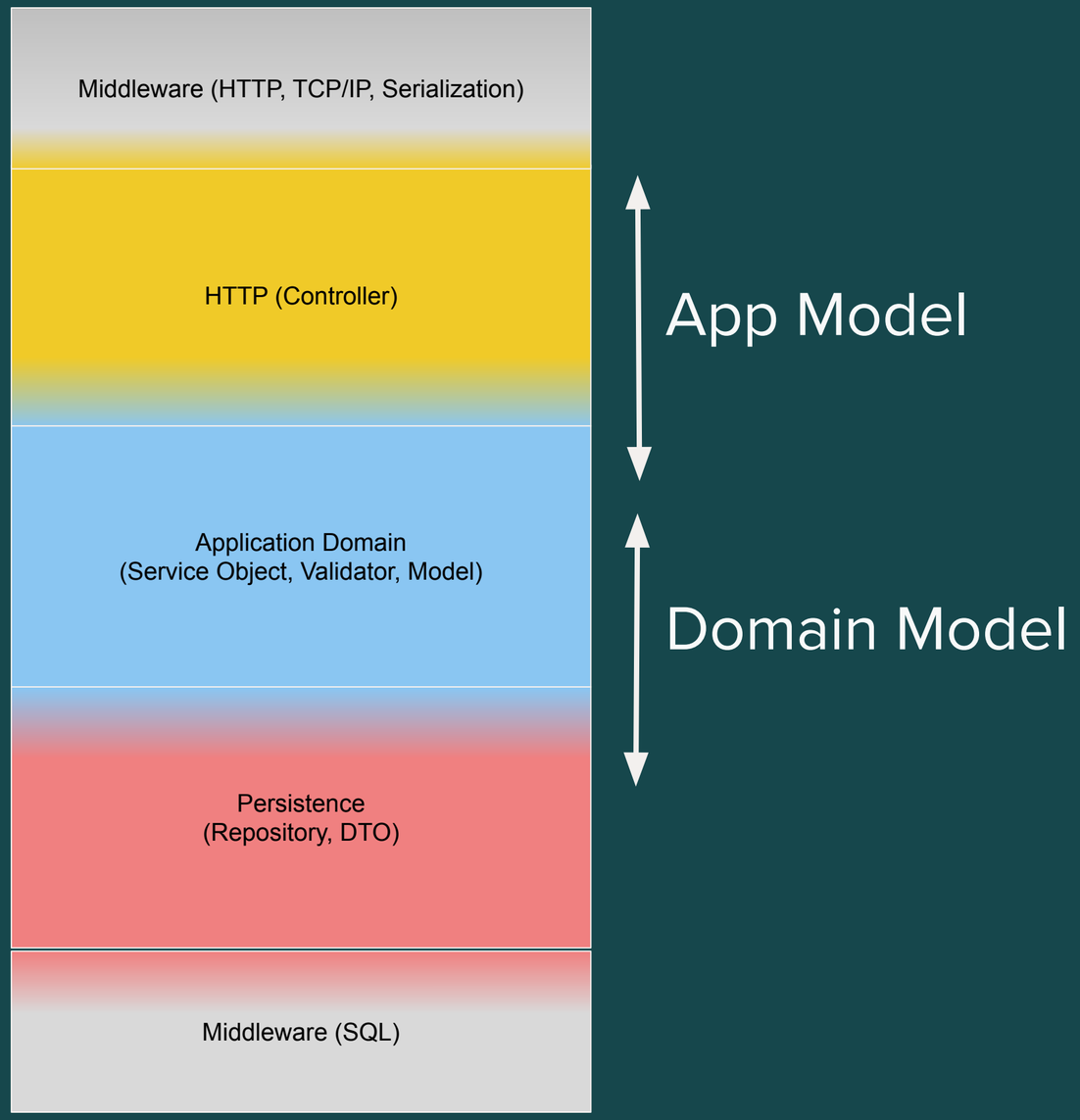
Hash

```java
public class ProductGoalsService {
    private final ProductGoalsRepository productGoalsRepository;

    public ProductGoal createProductGoal(
            User user,
            ProductGoalCreateParams createParams) {
        validate(createParams);

        var productGoal = createFromRequest(user.getAccountId(), createParams);
        return productGoalsRepository.save(productGoal);
    }

    private ProductGoal createFromRequest(
            long accountId,
            ProductGoalCreateParams createRequest) {
        return new ProductGoal(
                accountId,
                createRequest.getAssigneeId(),
                createRequest.getProductId(),
                createRequest.getProviderId(),
                createRequest.getQuota(),
                createRequest.getStartDate(),
                createRequest.getEndDate()
        );
    }
}

@Table(name = "product_goals")
@Entity
public class ProductGoal {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "account_id")
    private long accountId;

    @Column(name = "assignee_id")
    private long assigneeId;

    @Embedded
    private ProductGoalDimensions dimensions;

    ...
}
```

Rails


Middleware (HTTP, TCP/IP, Serialization)
HTTP (Controller, Sanitizer)
Application Domain
(Service Object, Model)
+
Persistence
(Model)
Middleware (SQL)
Hash
Model

Java


Middleware (HTTP, TCP/IP, Serialization)
HTTP (Controller)
Application Domain
(Service Object, Validator, Model)
Persistence
(Repository, DTO)
Middleware (SQL)
App Model
Domain Model

Is this all wrong?

not really...
until things are small

Coming back to

zendesk

Email4Big

Context

- Email is one of key features of Sell
- Mailman developed over **5 years**
- Mobile Apps have offline capabilities
- Lots and lots of small objects to sync to mobile

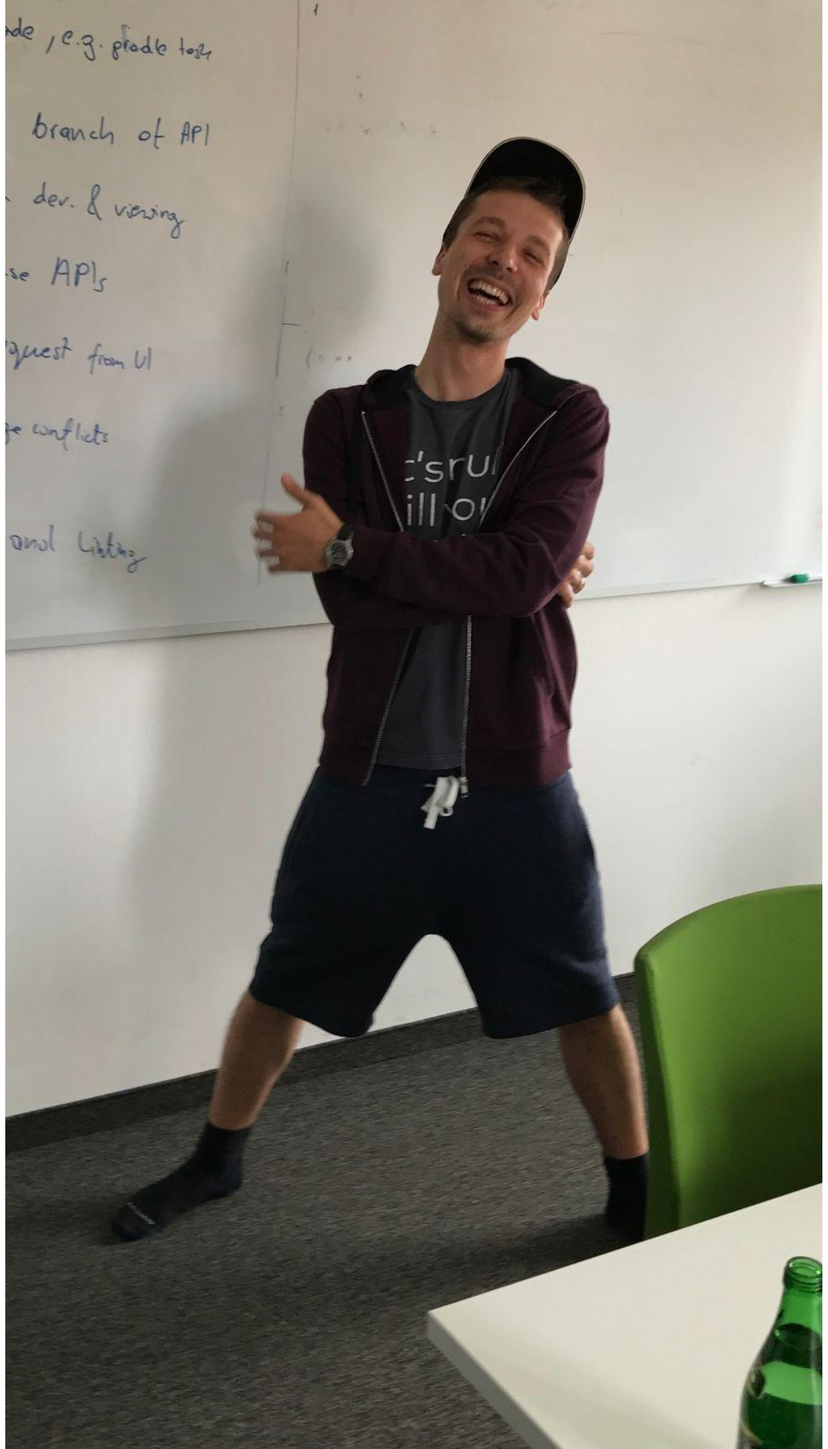
Problem

- Mobile sync **performance** on big accounts

Goals

- Fix performance issues
- **Encapsulate complexity to unlock future refactoring**

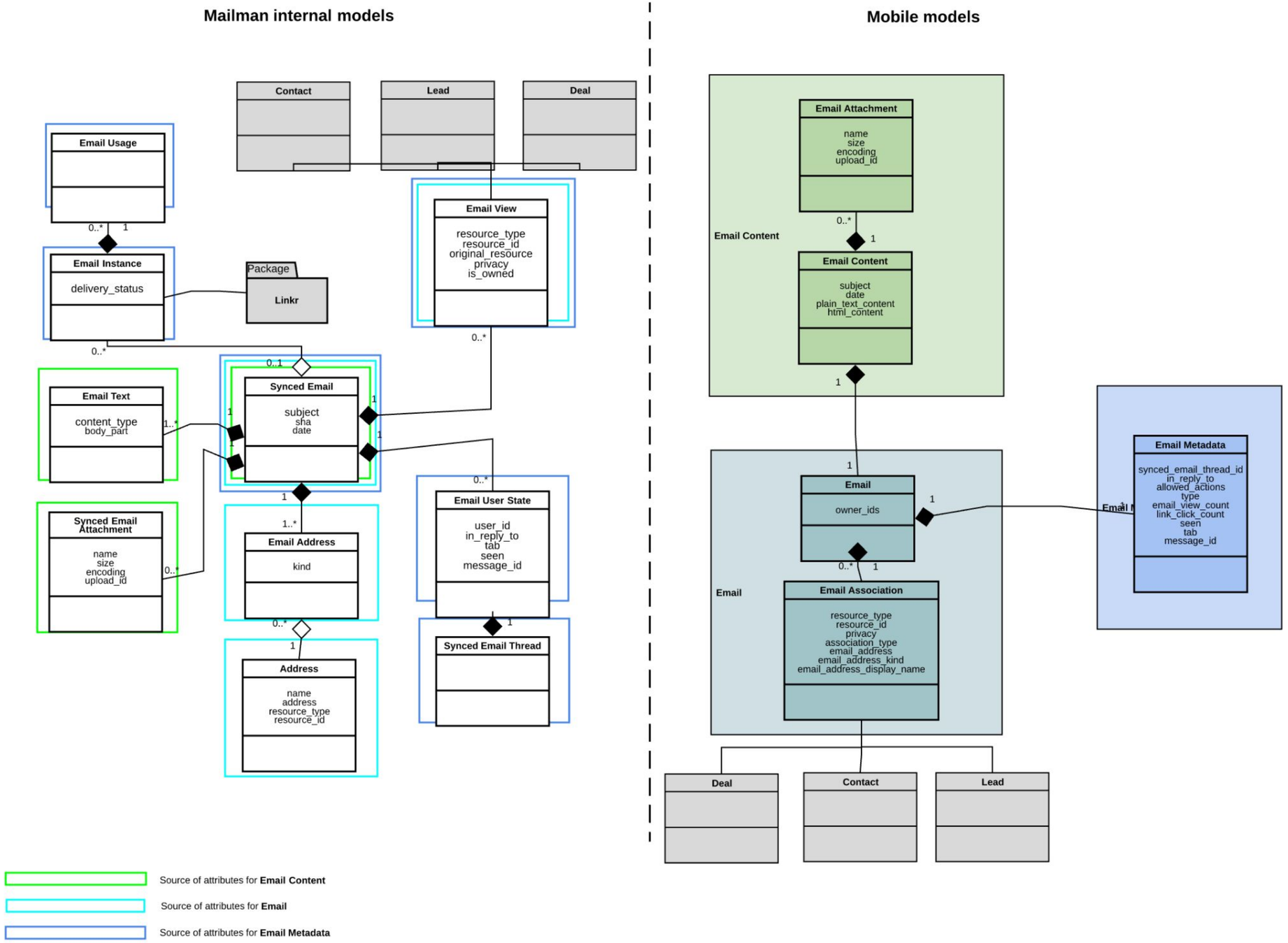
So I started...



Domain Modeling

- Ubiquitous Language
- Entities, Aggregates
- **Anticorruption Layer**

Tim Porter, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons


Mailman internal models
Mobile models
Contact
Lead
Deal
Email Usage
Email Instance
delivery_status
Package
Linkr
Email View
resource_type
resource_id
original_resource
privacy
is_owned
Email Text
content_type
body_part
Synced Email
subject
sha
date
Synced Email Attachment
name
size
encoding
upload_id
Email Address
kind
Email User State
user_id
in_reply_to
tab
seen
message_id
Address
name
address
resource_type
resource_id
Synced Email Thread
Email Attachment
name
size
encoding
upload_id
Email Content
Email Content
subject
date
plain_text_content
html_content
Email
Email
owner_ids
Email Association
resource_type
resource_id
privacy
association_type
email_address
email_address_kind
email_address_display_name
Email Metadata
synced_email_thread_id
in_reply_to
allowed_actions
type
email_view_count
link_click_count
seen
tab
message_id
Deal
Contact
Lead
Source of attributes for Email Content
Source of attributes for Email
Source of attributes for Email Metadata

Naming

Service Object

Command

“There are only two hard things in Computer Science: cache invalidation and ***naming things****”*

Phil Karlton

Naming

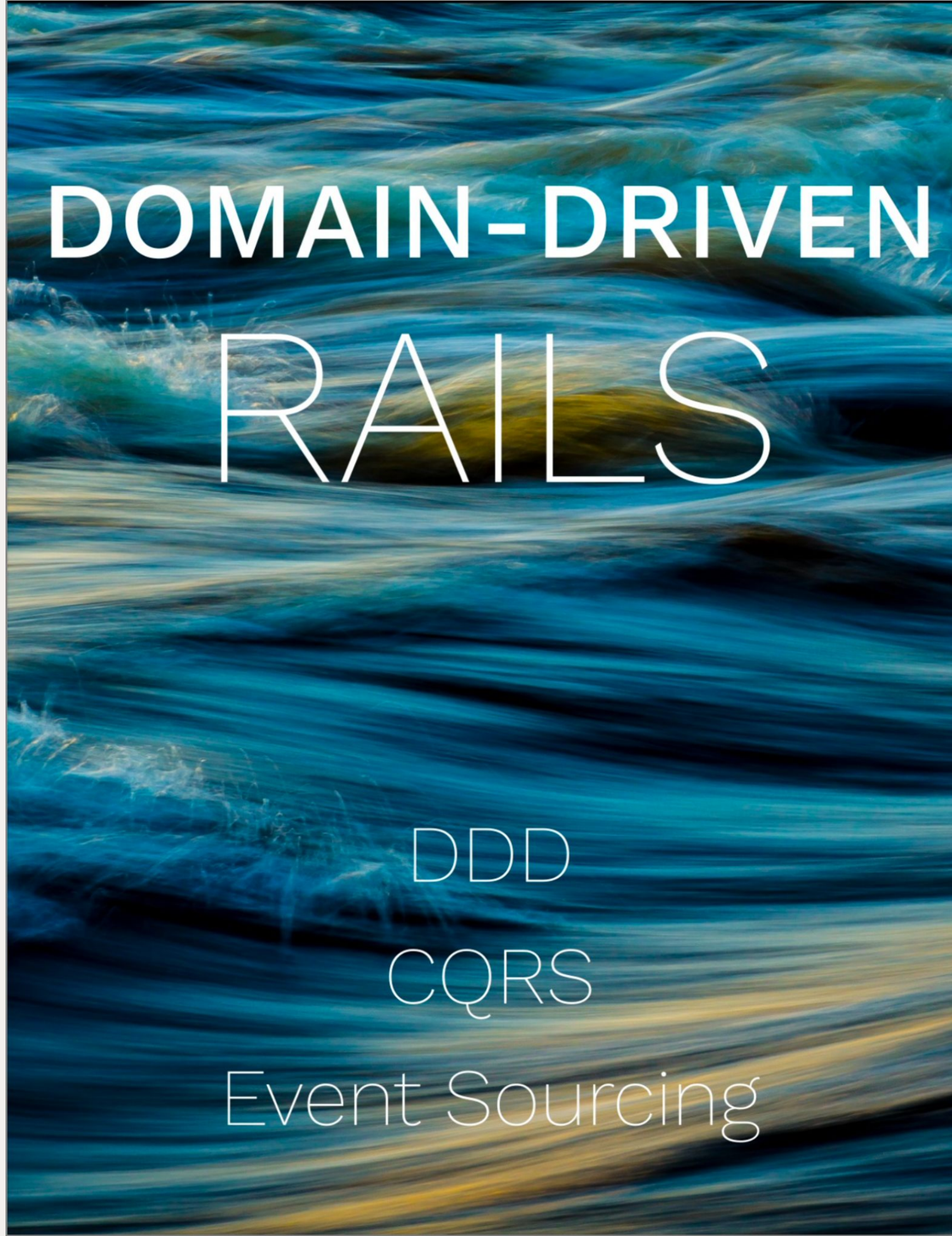
```ruby
class VisitBuilder < Struct.new(:user, :params)
 attr_reader :visit

 def create
   build_visit
   validate
   in_transaction do
     save
   end
   visit
 end

 def build_visit
   @visit = Visit.new(params)
   @visit.creator_id = user.id
   @visit.permissions_holder_id = user.id
   @visit.account_id = user.account_id
 end

 def validate
   visit.valid?
   raise ActiveRecord::RecordInvalid.new(visit) unless visit.errors.empty?
 end

 def save
   visit.save!
 end
end
```

```ruby
class VisitBuilder < Struct.new(:user, :visit_attributes)
 attr_reader :visit

 def create
   build_visit
   validate
   in_transaction do
     save
   end
   visit
 end

 def build_visit
   @visit = Visit.new(visit_attributes)
   @visit.creator_id = user.id
   @visit.permissions_holder_id = user.id
   @visit.account_id = user.account_id
 end

 def validate
   visit.valid?
   raise ActiveRecord::RecordInvalid.new(visit) unless visit.errors.empty?
 end

 def save
   visit.save!
 end
end
```

Domain-Driven Rails

DDD, CQRS, Event Sourcing

Robert Pankowecki & Arkency Team



Command

```
CreateVisitCommand = Struct.new(
  :user,
  :visited_at,
  ...
  :summary,
  :rep_address) do

  def user_id
    user.id
  end

  def account_id
    user.account_id
  end

  def valid?
    ...
  end
end

def create
  ...
  data = request.params[:data]
  create_visit_command = CreateVisitCommand.new(
    user: request.user,
    visited_at: data.fetch(:visited_at),
     ...
    rep_address: data.fetch(:rep_address),
  )
  visit = VisitBuilder.create(create_visit_command)
  response.item_created(Visit::V1Representer.new(visit))
rescue ActiveRecord::RecordInvalid => e
  response.model_errors(e.record)
end
```

```
class VisitBuilder < Struct.new(:create_visit_command)
  attr_reader :visit

  def create
    validate
    build_visit
    in_transaction do
      save
    end
    visit
  end

  def build_visit
    @visit = Visit.new
    @visit.visited_at = create_visit_command.visited_at,
    @visit.resource_type = create_visit_command.resource_type,
    @visit.resource_id = create_visit_command.resource_id,
    @visit.outcome_id = create_visit_command.outcome_id,
    @visit.rep_latitude = create_visit_command.rep_latitude,
    @visit.resource_address = create_visit_command.resource_address,
    @visit.rep_location_verification_status =
       create_visit_command.rep_location_verification_status,
    @visit.summary = create_visit_command.summary,
    @visit.rep_address = create_visit_command.rep_address,
    @visit.creator_id = create_visit_command.user_id
    @visit.permissions_holder_id = create_visit_command.user_id
    @visit.account_id = create_visit_command.account_id
  end

  def validate
    create_visit_command.valid?
  end
```

~~Hash~~

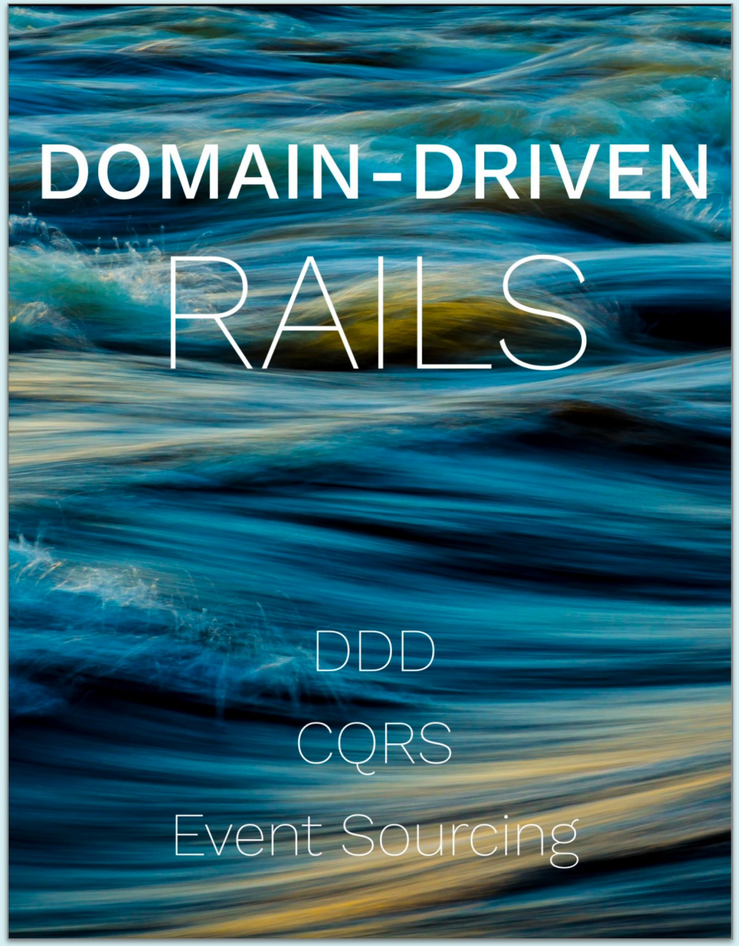
Object-Oriented Programming!

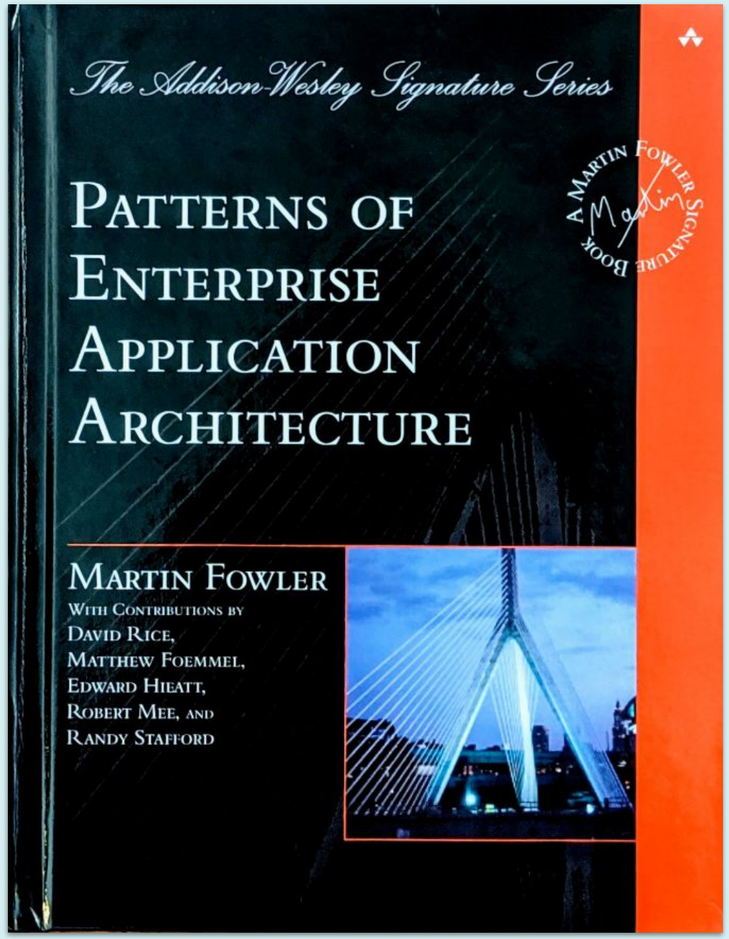
Constraints!

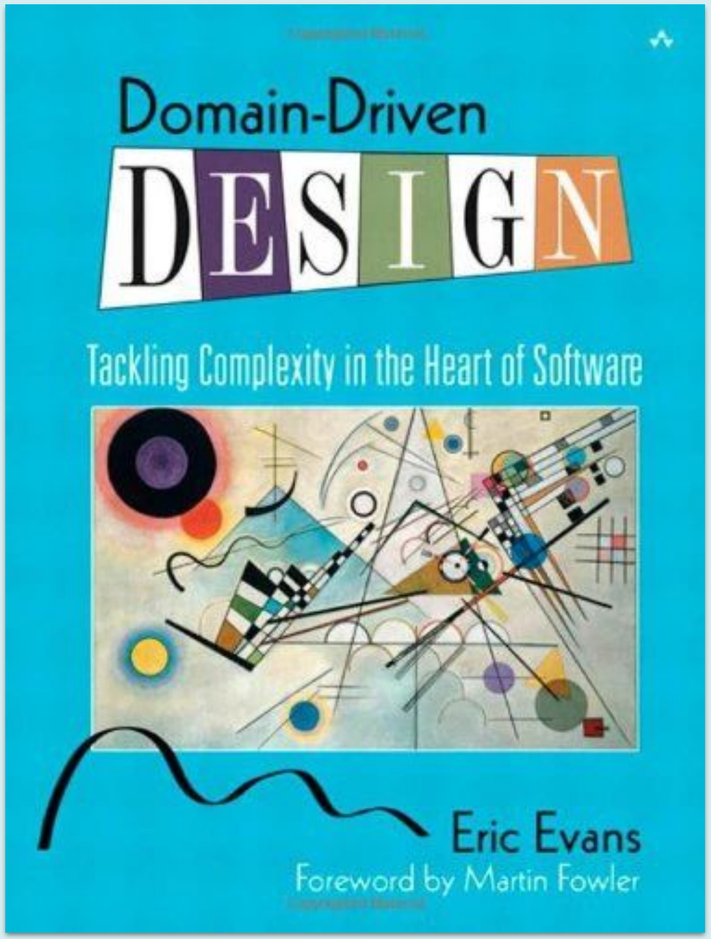
- Names
- Classes
- Pre/Post-conditions, Invariants
- Design Patterns
  - Command
  - Service Object
- Architecture Patterns

“Programs must be written for people to read, and only incidentally for machines to execute.”

Harold Abelson
Structure and Interpretation of Computer Programs


DOMAIN-DRIVEN
RAILS
DDD
CQRS
Event Sourcing


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE
MARTIN FOWLER
WITH CONTRIBUTIONS BY
DAVID RICE,
MATTHEW FOEMMEL,
EDWARD HIEATT,
ROBERT MEE, AND
RANDY STAFFORD


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

www.zendesk.com/sell/

Make a big deal of it

Sales CRM to maximize productivity, pipeline visibility, and revenue for sales teams

Try Sell for free

zendesk sell

Overview Features Pricing Contact us Start free trial View demo

SALES CRM SOFTWARE

A modern sales CRM to help you accelerate revenue

Q&A

zendesk