

WHEN EASY IS HARD



15 June 2015





Marek Matulka

Software Engineer
at Sensio**Labs**UK



@super_marek

The Inviqa Group comprises
four brands all specialising
in enterprise open source

INVIQA™

Sensio**Labs**
UK

.session

iKOS



Symphony Live

LONDON 2015

SEPTEMBER 17-18

<http://london2015.live.symphony.com/>



Take it easy!

Framework



Convenience

Prologue

[Release Notes](#)
[Upgrade Guide](#)
[Contribution Guide](#)

Setup

[Installation](#)
[Configuration](#)
[Homestead](#)

The Basics

[Routing](#)
[Middleware](#)
[Controllers](#)
[Requests](#)
[Responses](#)
[Views](#)

Architecture

Foundations

[Service Providers](#)
[Service Container](#)
[Contracts](#)
[Facades](#)
[Request Lifecycle](#)
[Application Structure](#)


a **SensioLabs** Product

[DOWNLOAD](#)
[What is Symfony?](#)
[Get started](#)
[Documentation](#)
[Community](#)
[Marketplace](#)
[Jobs](#)
[Business Solutions](#)
[News](#)

The Book

The Cookbook

The Components

Symfony Best Practices

Reference

Index

Contributing

Bundles

Symfony CMF

Docs for symfony 1.x

Trainings

[Job](#) [ViscaWeb](#)

Barcelona, Spain

Full stack web developer - Symfony 2

[jobs.sensiolabs.com](#)

[Master Symfony2 fundamentals](#)

Be trained by SensioLabs experts (2 to

Learn Symfony


[Read the book →](#)

The Symfony Book

Prepared by the core team, this is the reference that covers the needs for every kind of Symfony developer.

[Controllers](#)
[Testing](#)
[HTTP Cache](#)
[Routing](#)
[Forms](#)
[Translation](#)
[Templating](#)
[Validation](#)
[Services](#)
[Doctrine](#)
[Security](#)
[Performance](#)

The Symfony Cookbook

A collection of tutorials explaining how to solve the most recurrent problems faced by Symfony developers.

[Read all tutorials →](#)


Generators

[Controller Generator](#)

[Crud Generator](#)

[Form Generator](#)

[Model Generator](#)

[Module Generator](#)

Model Generator

This generator generates a model class for the specified database table.

Fields with * are required. Click on the highlighted fields to edit them.

Table Prefix

tbl_

Table Name *

tbl_user

Model Class *

User

Base Class *

CActiveRecord

Model Path *

application.models

Code Template *

default (D:\yii\framework\yii\generators\model\templates\default)

Code File

[models\User.php](#)

```
/**
 * Displays a particular model.
 * @param integer $id the ID of the model to be displayed
 */
public function actionView($id)
{
    $this->render('view',array(
        'model'=>$this->loadModel($id),
    ));
}

/**
 * Creates a new model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 */
public function actionCreate()
{
    $model=new User;

    // uncomment the following line if AJAX validation is needed
    // $this->performAjaxValidation($model);

    if(isset($_POST['User']))
    {
        $model->attributes=$_POST['User'];
        if($model->save())
            $this->redirect(array('view','id'=>$model->id));
    }

    $this->render('create',array(
        'model'=>$model,
    ));
}
```

```
<?php namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

class UserController extends Controller {

    /**
     * Show the profile for the given user.
     *
     * @param int $id
     * @return Response
     */
    public function showProfile($id)
    {
        return view('user.profile', ['user' => User::findOrFail($id)]);
    }
}
```

```
<?php

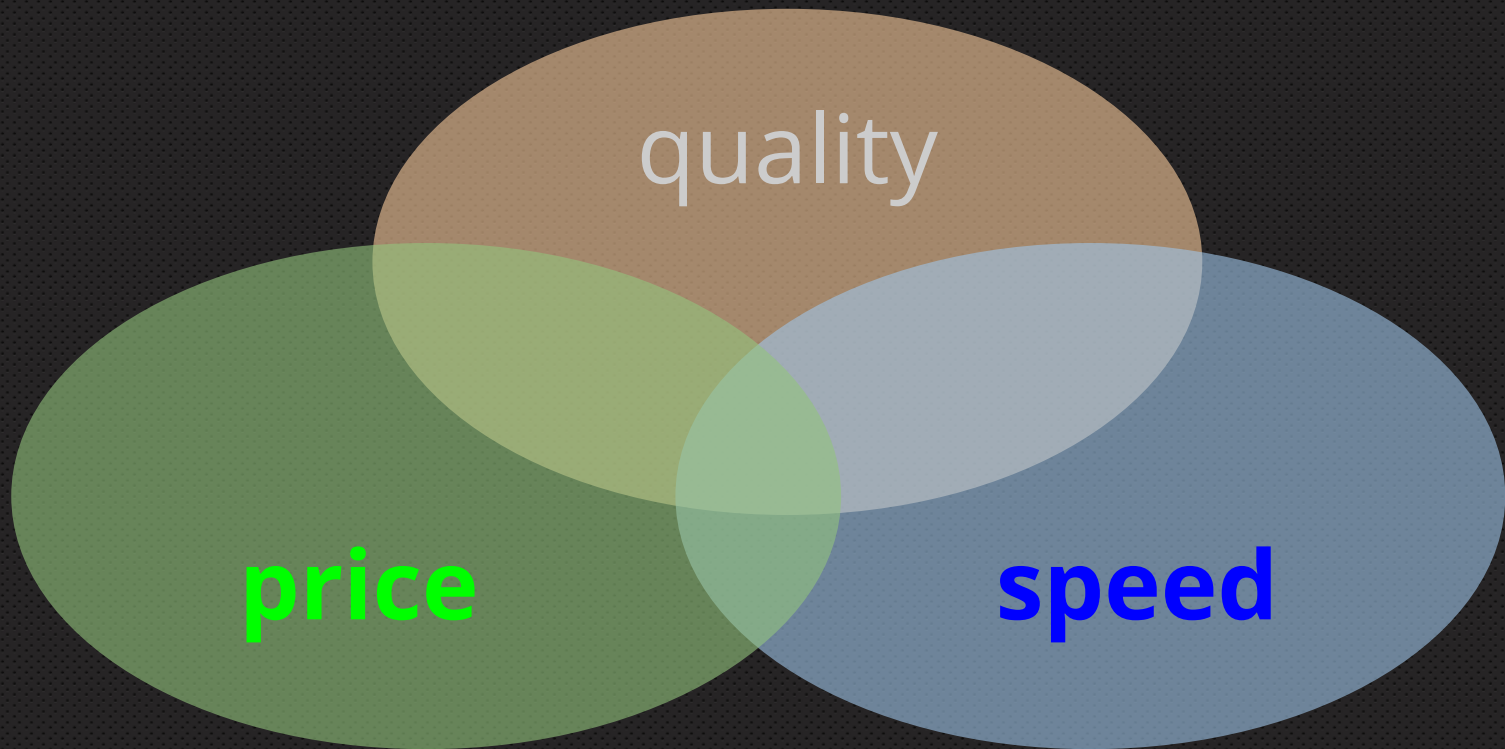
namespace Acme\ConferenceBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Response;

class RegistrationController extends Controller
{
    public function registerAction($id)
    {
        $conference = $this->getDoctrine()
            ->getRepository('AcmeConferenceBundle:Conference')
            ->find($id);
        $this->get('acme_conference.enrolment.service')
            ->enrol($conference, $this->getUser()) {

        return new Response('Success!');
    }
}
```

Cost



Phase 1 completed!





Let's grow!

New features

```

<?php

namespace Acme\ConferenceBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class RegistrationController extends Controller
{
    public function registerAction($id)
    {
        $conference = $this->getDoctrine()
            ->getRepository('AcmeConferenceBundle:Conference')->find($id);
        if (
            $enrolment = $this->get('acme_conference.enrolment.service')
                ->enrol($conference, $this->getUser())
        ) {
            $em = $this->getDoctrine()->getEntityManager();
            $em->persist($enrolment);
            $em->flush();
            $this->get('acme_conference.notification.service')
                ->notify($this->getUser(), 'success');
            return $this->redirect('registration_successful')
        }
        return $this->redirect('registration_failed');
    }
}

```



A still from the movie Toy Story showing Woody and Buzz Lightyear. Woody is on the left, looking slightly concerned. Buzz is on the right, looking excited and holding up his right hand with three fingers spread. The background is a simple indoor setting with a door and some toys on the floor.

BROKEN CODE

BROKEN CODE EVERYWHERE

I DIDN'T CHOOSE THE BUG LIFE



THE BUG LIFE CHOOSE ME



Yummy spaghetti code!

...and there are **no tests!**



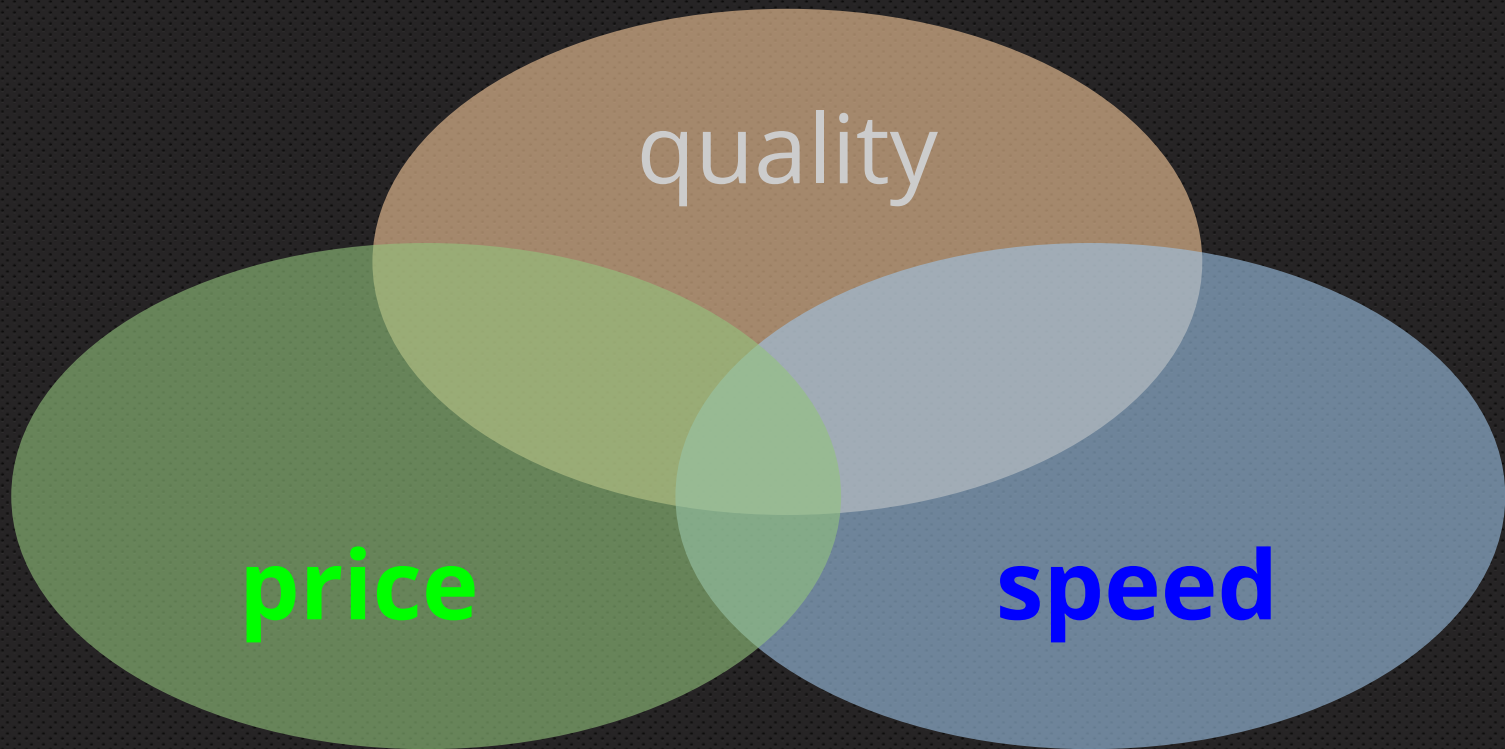
No one wants to touch it

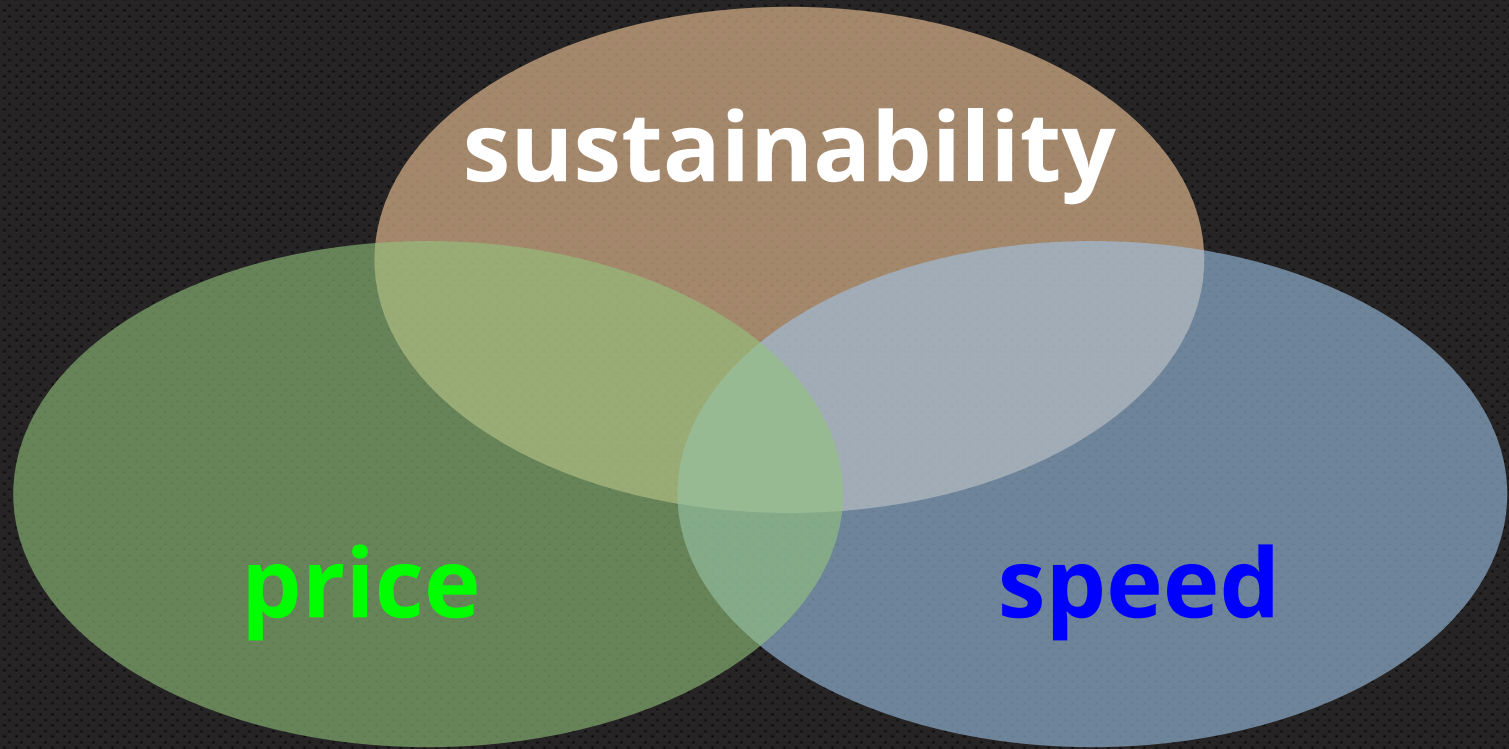
Let's rewrite it!





not so happy customer







What has happened?

What has happened?

- too close to the framework

What has happened?

- too close to the framework
- no separation of concerns

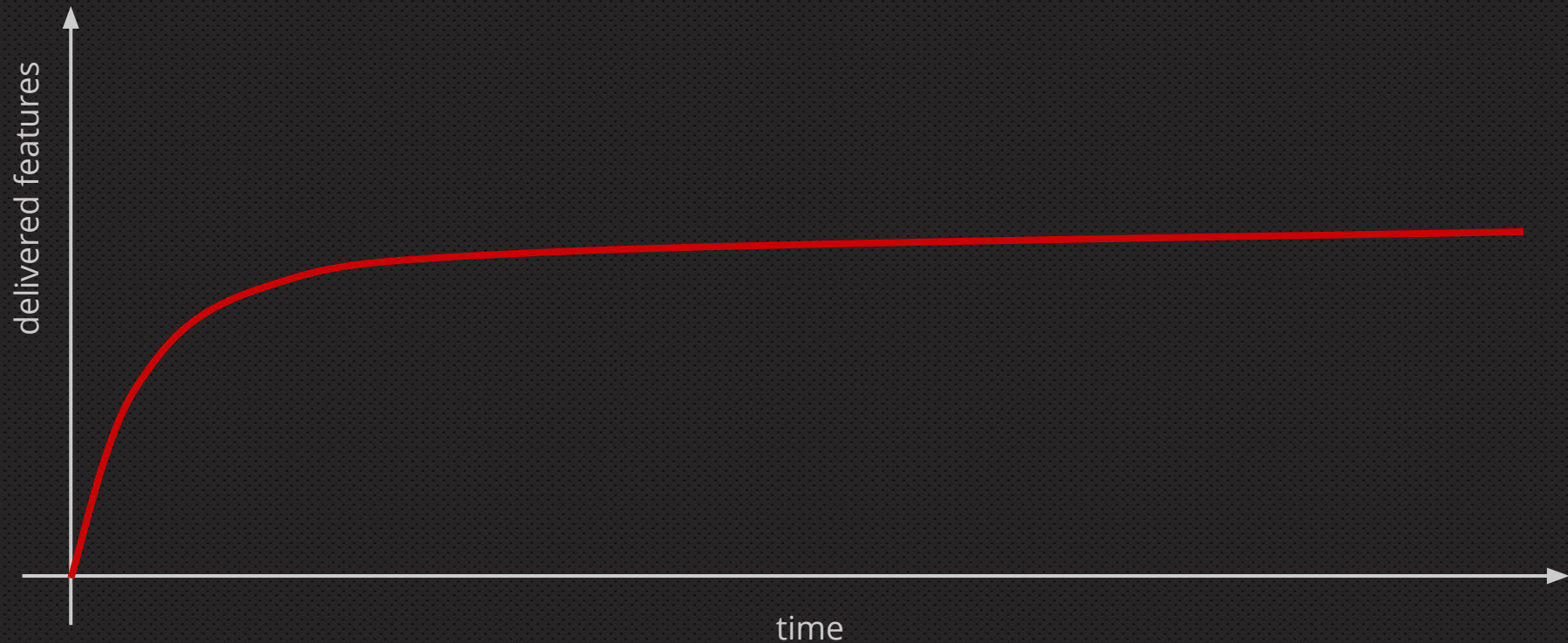
What has happened?

- too close to the framework
- no separation of concerns
- mixed layers

What has happened?

- too close to the framework
- no separation of concerns
- mixed layers
- no tests!

Features delivered over time





How to avoid?

Forget the framework!

Decouple your code

Decoupled code

- improves readability

Decoupled code

- improves readability
- can be reused

Decoupled code

- improves readability
- can be reused
- easy to maintain

Decoupled code

- improves readability
- can be reused
- easy to maintain
- easy to change



Let's do it!

Avoid hidden dependencies

*Global scope is a **smell**!*

```
class RegistrationController
{
    public function registerAction($id)
    {
        $learner = Session::getUser();
        $conference = Conference::findOrFail($id);

        Enrolment::enrollLearnerInConference($learner, $conference);

        Mail::send('emails.welcome', array('key' => 'value'),
            function($message) use ($learner, $conference)
            {
                $message->to($learner->email(), $learner->name())
                    ->subject('Thanks for registering for '.$conference->name().'!');
            }
        );
    }
}
```

Inject only what you need

```
class RegistrationController
```

```
{
```

```
    public function __construct(
```

```
        Enrolment $enrolment,
```

```
        Learner $learner,
```

```
        Notifier $notifier
```

```
    ) {
```

```
        $this->enrolment = $enrolment;
```

```
        $this->learner = $learner;
```

```
        $this->notifier = $notifier;
```

```
    }
```

```
    public function registerAction(Conference $conference)
```

```
    {
```

```
        $enrolment = $this->enrolment->enrolLearnerInConference($this->learner, $conference);
```

```
        $this->notifier->successfulEnrolment($enrolment);
```

```
    }
```

```
}
```

Don't mix layers

```
class RegistrationController
{
    public function __construct(
        EnrolmentHandler $enrolmentHandler,
        Learner $learner
    ) {
        $this->enrolmentHandler = $enrolmentHandler;
        $this->learner = $learner;
    }

    public function registerAction(Conference $conference)
    {
        $this->enrolmentHandler->handle(
            new EnrolLearnerInConference($this->learner, $conference);
        );
    }
}
```

```
class RegistrationController
{
    public function __construct(
        EnrolmentHandler $enrolmentHandler,
        Learner $learner
    ) {
        $this->enrolmentHandler = $enrolmentHandler;
        $this->learner = $learner;
    }

    public function registerAction(Conference $conference)
    {
        $this->enrolmentHandler->handle(
            new EnrolLearnerInConference($this->learner, $conference);
        );
    }
}
```

```
class RegistrationController
{
    public function registerAction($id)
    {
        $learner = Session::getUser();
        $conference = Conference::findOrFail($id);

        Enrolment::enrollLearnerInConference($learner, $conference);

        Mail::send('emails.welcome', array('key' => 'value'),
            function($message) use ($learner, $conference)
            {
                $message->to($learner->email(), $learner->name())
                    ->subject('Thanks for registering for '.$conference->name().'!');
            }
        );
    }
}
```

```
class RegistrationController
{
    public function __construct(
        EnrolmentHandler $enrolmentHandler,
        Learner $learner
    ) {
        $this->enrolmentHandler = $enrolmentHandler;
        $this->learner = $learner;
    }

    public function registerAction(Conference $conference)
    {
        $this->enrolmentHandler->handle(
            new EnrolLearnerInConference($this->learner, $conference);
        );
    }
}
```

```
class EnrolmentHandler
{
    public function __construct(EnrolmentRepository $enrolmentRepository, Mailer $mailer)
    {
        $this->repository = $enrolmentRepository;
        $this->mailer = $mailer;
    }

    public function handle(EnrolLearnerInConference $enrolmentCommand)
    {
        $enrolment = $enrolmentCommand->execute();
        $this->repository>save($enrolment);
        $this->mailer->send('emails.welcome', array('key' => 'value'),
            function($message) use ($enrolment) {
                $message->to($enrolment->learner()->email(), $enrolment->learner()->name())
                    ->subject('Thanks for registering for '.$enrolment->conference->name().'!');
            });
    }
}
```

Invert dependencies

```
class EnrolmentHandler
{
    public function __construct(EnrolmentRepository $enrolmentRepository, Mailer $mailer)
    {
        $this->repository = $enrolmentRepository;
        $this->mailer = $mailer;
    }

    public function handle(EnrolLearnerInConference $enrolmentCommand)
    {
        $enrolment = $enrolmentCommand->execute();
        $this->repository>save($enrolment);
        $this->mailer->send('emails.welcome', array('key' => 'value'),
            function($message) use ($enrolment) {
                $message->to($enrolment->learner()->email(), $enrolment->learner()->name())
                    ->subject('Thanks for registering for '.$enrolment->conference->name().'!');
            });
    }
}
```

```
class EnrolmentHandler
{
    public function __construct(
        EnrolmentRepository $enrolmentRepository,
        EnrolmentNotifier $notifier
    ) {
        $this->repository = $enrolmentRepository;
        $this->notifier = $notifier;
    }

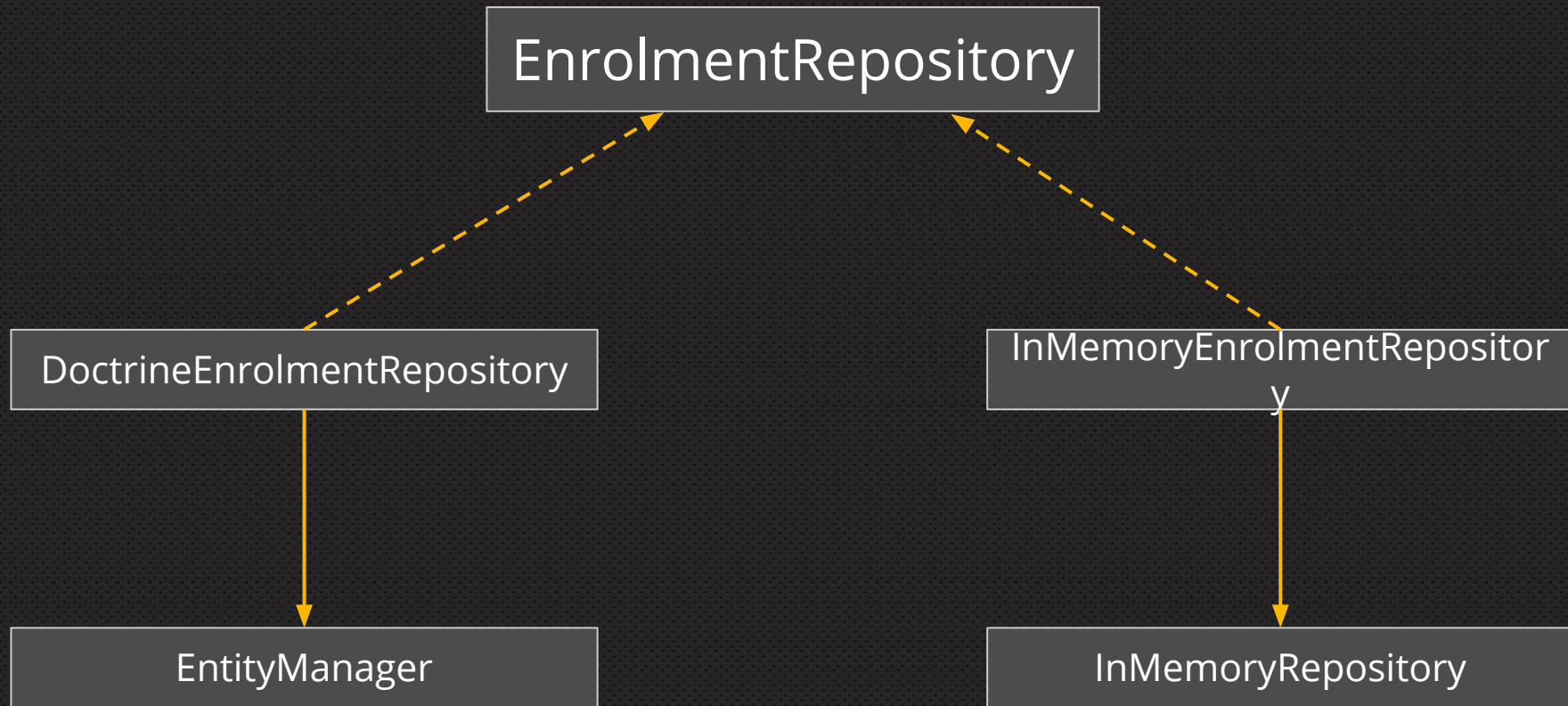
    public function handle(EnrolLearnerInConference $enrolmentCommand)
    {
        if ($enrolment = $enrolmentCommand->execute()) {
            $this->repository->save($enrolment);
            $this->notifier->successfulEnrolment($enrolment);
        }
    }
}
```

```
interface EnrolmentNotifier
{
    /**
     * @param Enrolment $enrolment
     *
     * @throws FaildToNotify
     */
    public function successfulEnrolment(Enrolment $enrolment);
}
```

```
interface EnrolmentRepository
{
    /**
     * @param integer $id
     * @return Enrolment
     */
    public function findOneById($id);

    /**
     * @param Enrolment $enrolment
     */
    public function save(Enrolment $enrolment);
}
```

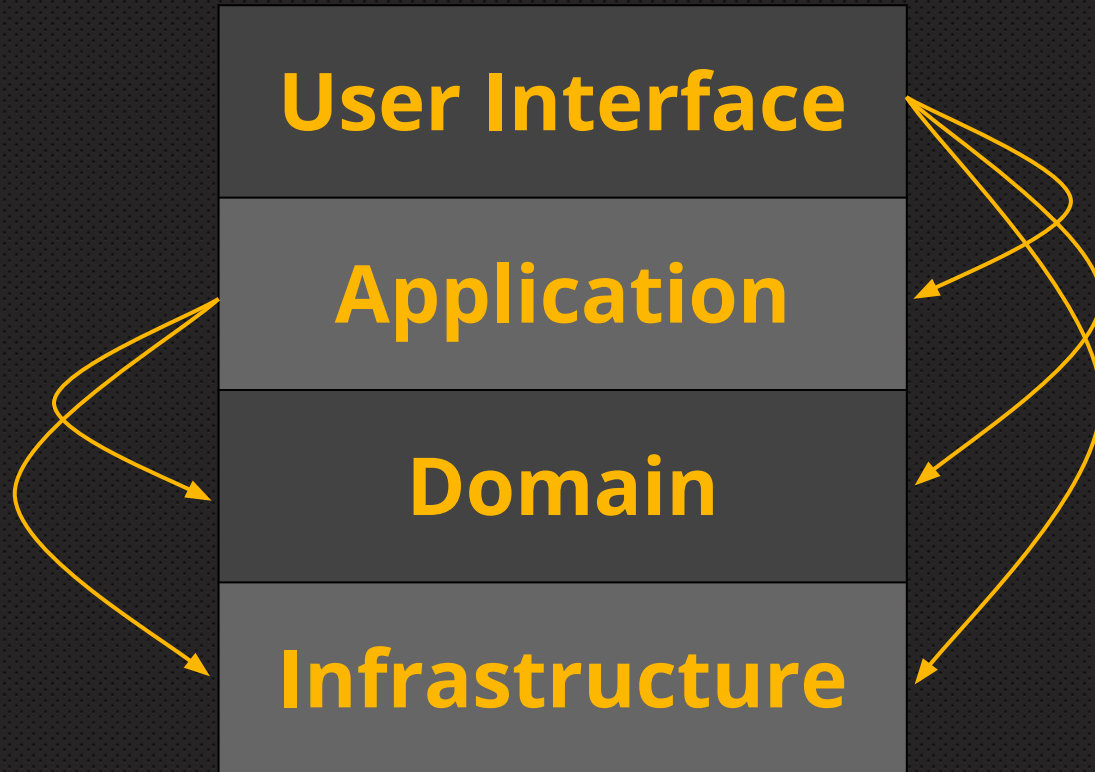
Dependency Inversion



Clean Code

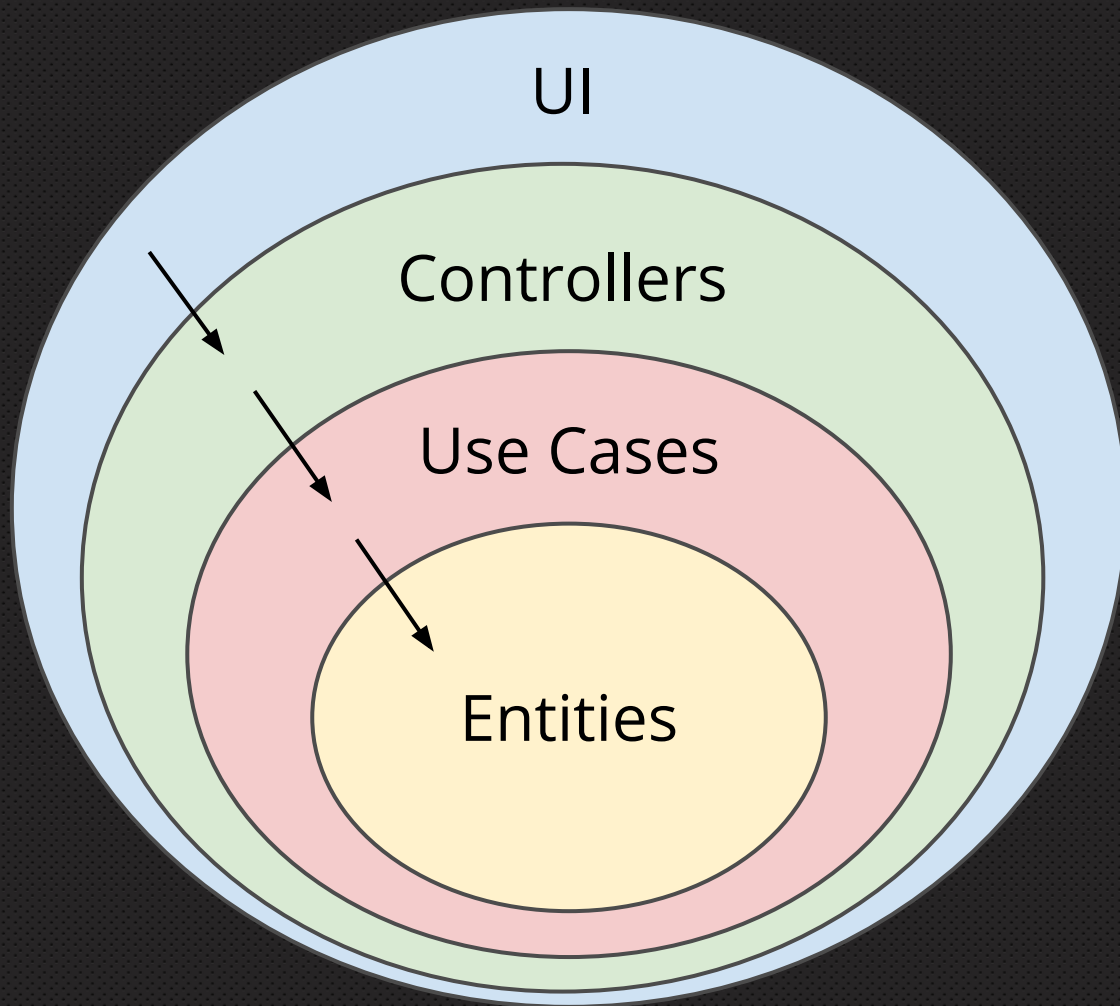


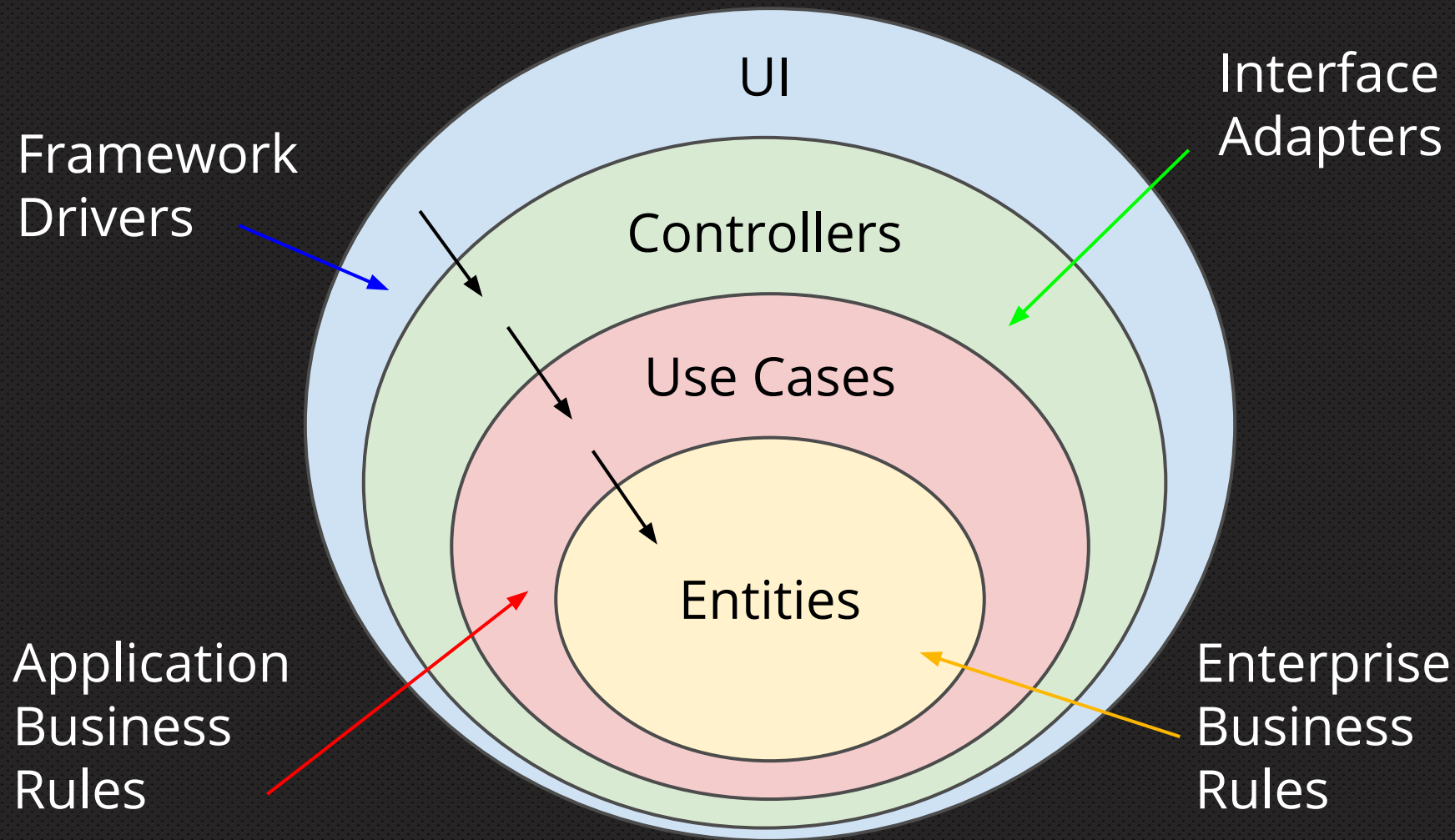
Layered Architecture



Clean Architecture

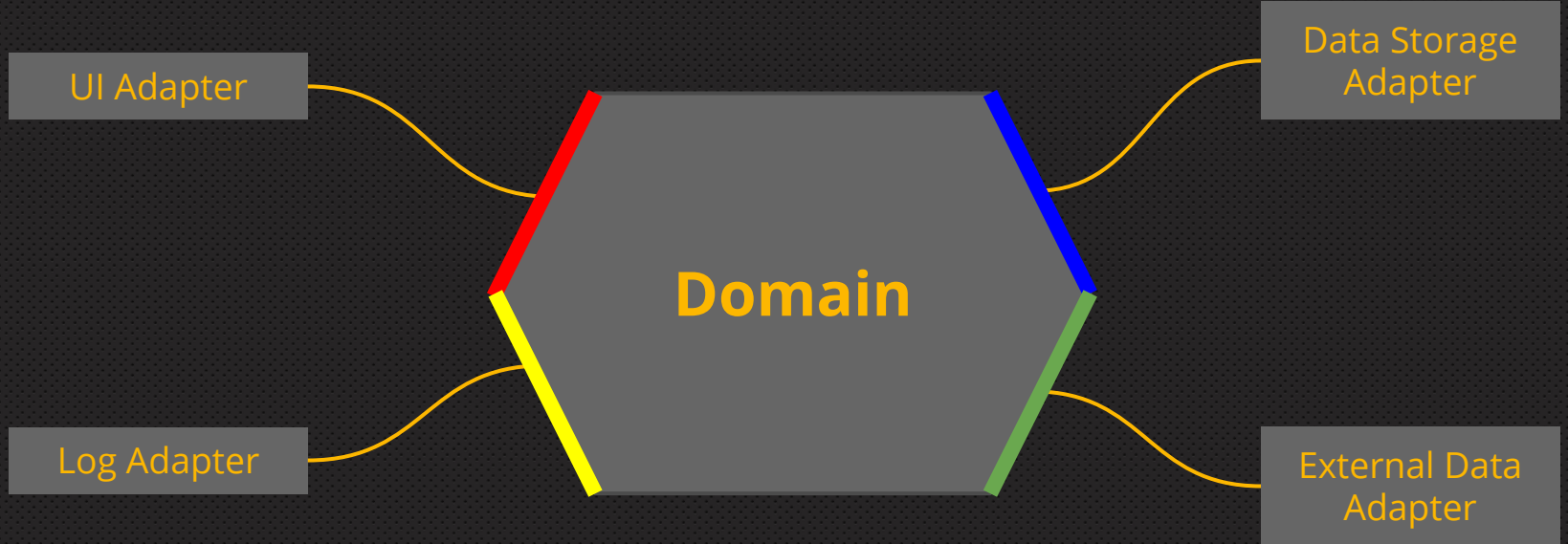
by Robert C. Martin

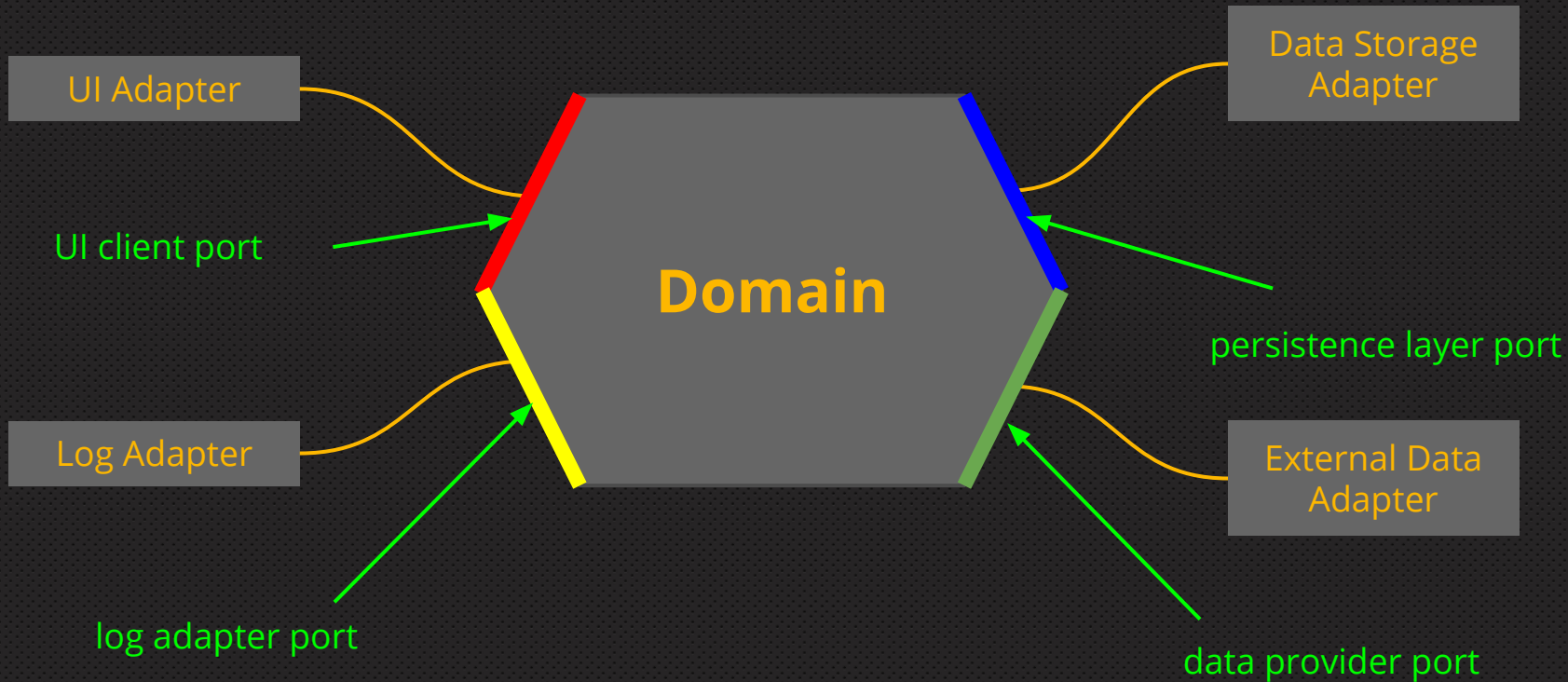


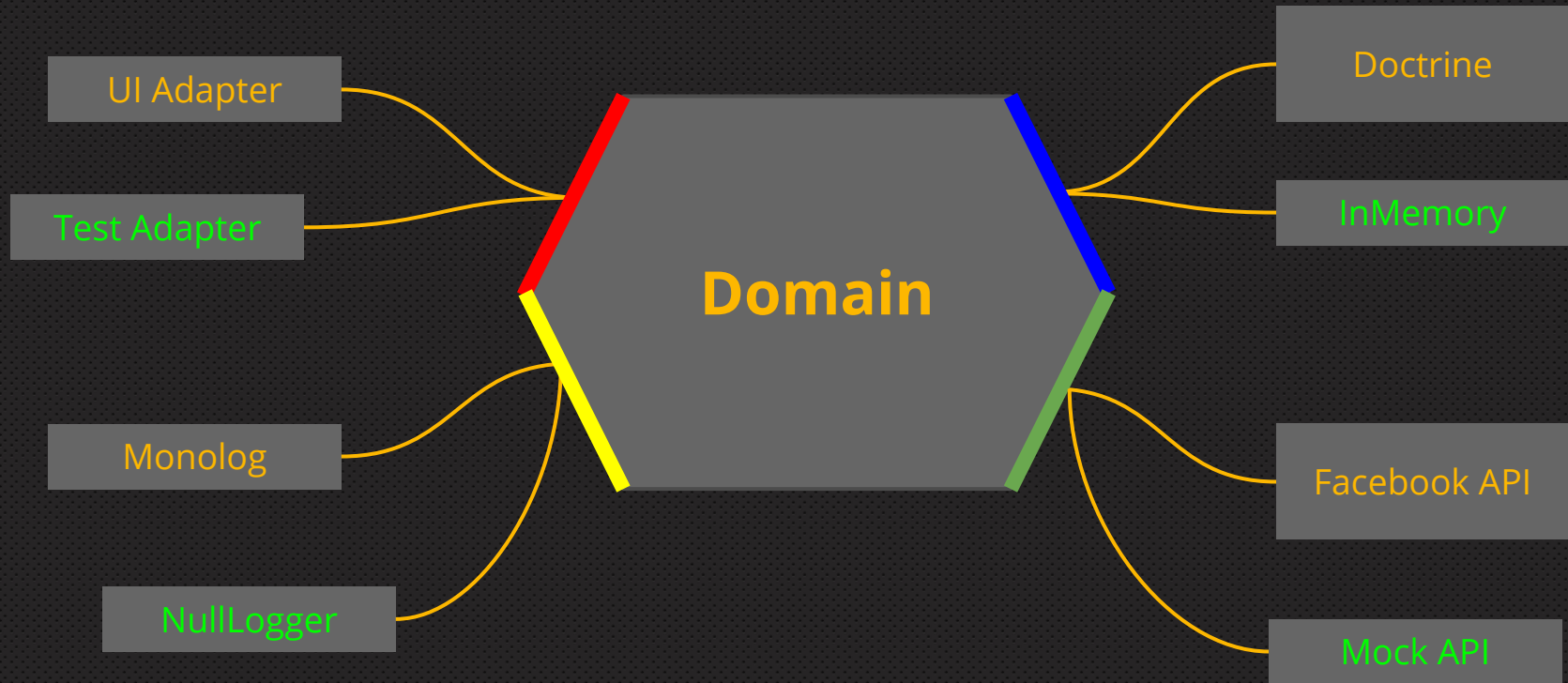


Hexagonal Architecture

aka Ports & Adapters







```
class EnrolmentHandler
{
    public function __construct(
        EnrolmentRepository $enrolmentRepository,
        EnrolmentNotifier $notifier
    ) {
        $this->repository = $enrolmentRepository;
        $this->notifier = $notifier;
    }

    public function handle(EnrolLearnerInConference $enrolmentCommand)
    {
        if ($enrolment = $enrolmentCommand->execute()) {
            $this->repository->save($enrolment);
            $this->notifier->successfulEnrolment($enrolment);
        }
    }
}
```

```
interface EnrolmentRepository
{
    /**
     * @param integer $id
     *
     * @return Enrolment
     */
    public function findOneById($id);

    /**
     * @param Enrolment $enrolment
     */
    public function save(Enrolment $enrolment);
}
```

```
class DoctrineEnrolmentRepository implements EnrolmentRepository
{
    public function __construct(EntityManager $entityManager)
    {
        $this->em = $entityManager;
    }

    public function findOneById($id)
    {
        return $this->em->find(Enrolment::class, $id);
    }

    public function save(Enrolment $enrolment)
    {
        $this->em->persist($enrolment);
        $this->em->flush($enrolment);
    }
}
```

```
class InMemoryEnrolmentRepository implements EnrolmentRepository
{
    public function __construct()
    {
        $this->repository = new InMemoryRepository();
    }

    public function findOneById($id)
    {
        return $this->repository->findById($id);
    }

    public function save(Enrolment $enrolment)
    {
        $this->repository->save($enrolment);
    }
}
```

Hard!

(five more classes to write!)



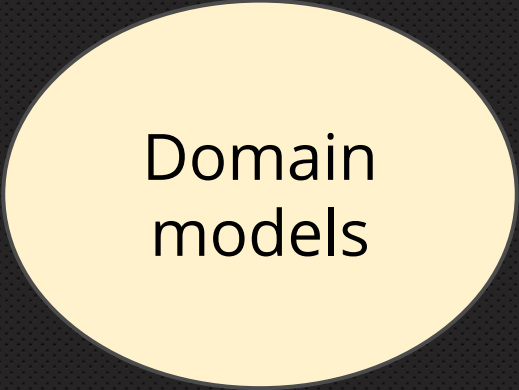
**AIN'T NOBODY
GOT TIME FOR
THAT!**

The background of the text is a dynamic action scene. It features a large dark ship, possibly a battleship, firing bright orange and red energy beams or missiles. In the foreground, there are smaller figures, possibly fighters or drones, and bright explosions or impacts on the ground or water, creating a sense of intense battle.

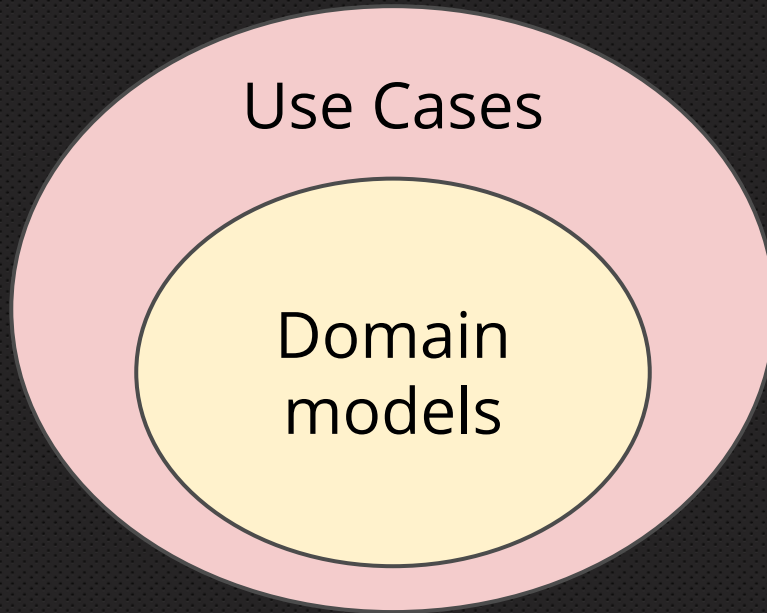


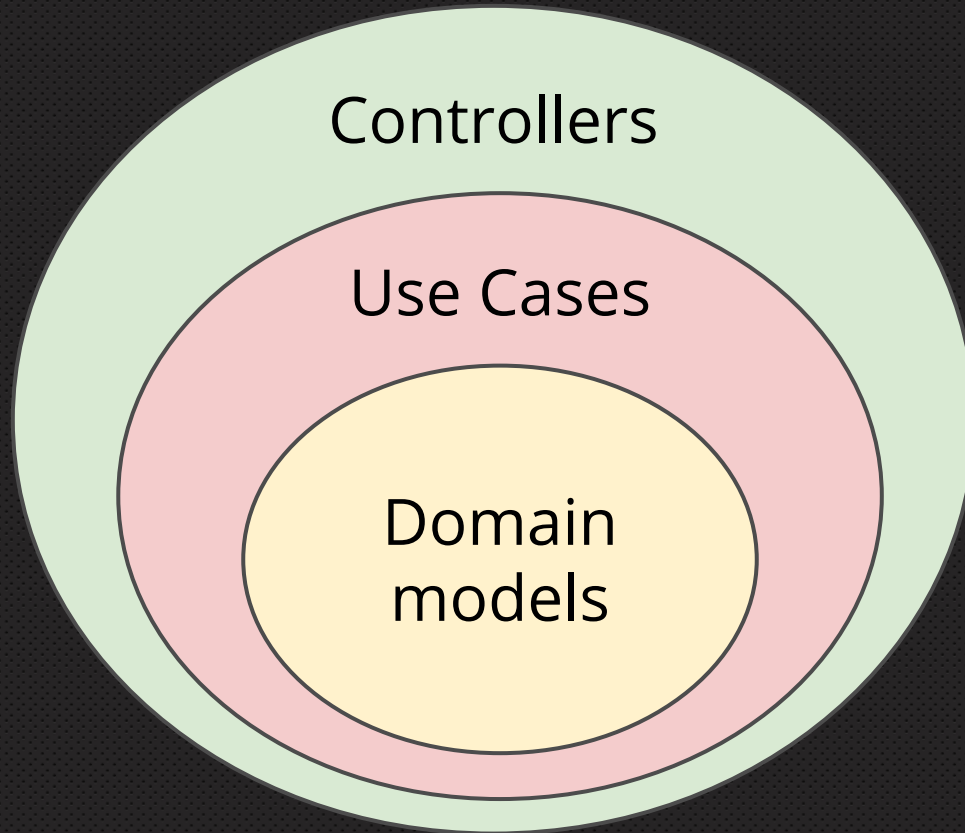
Why bother?

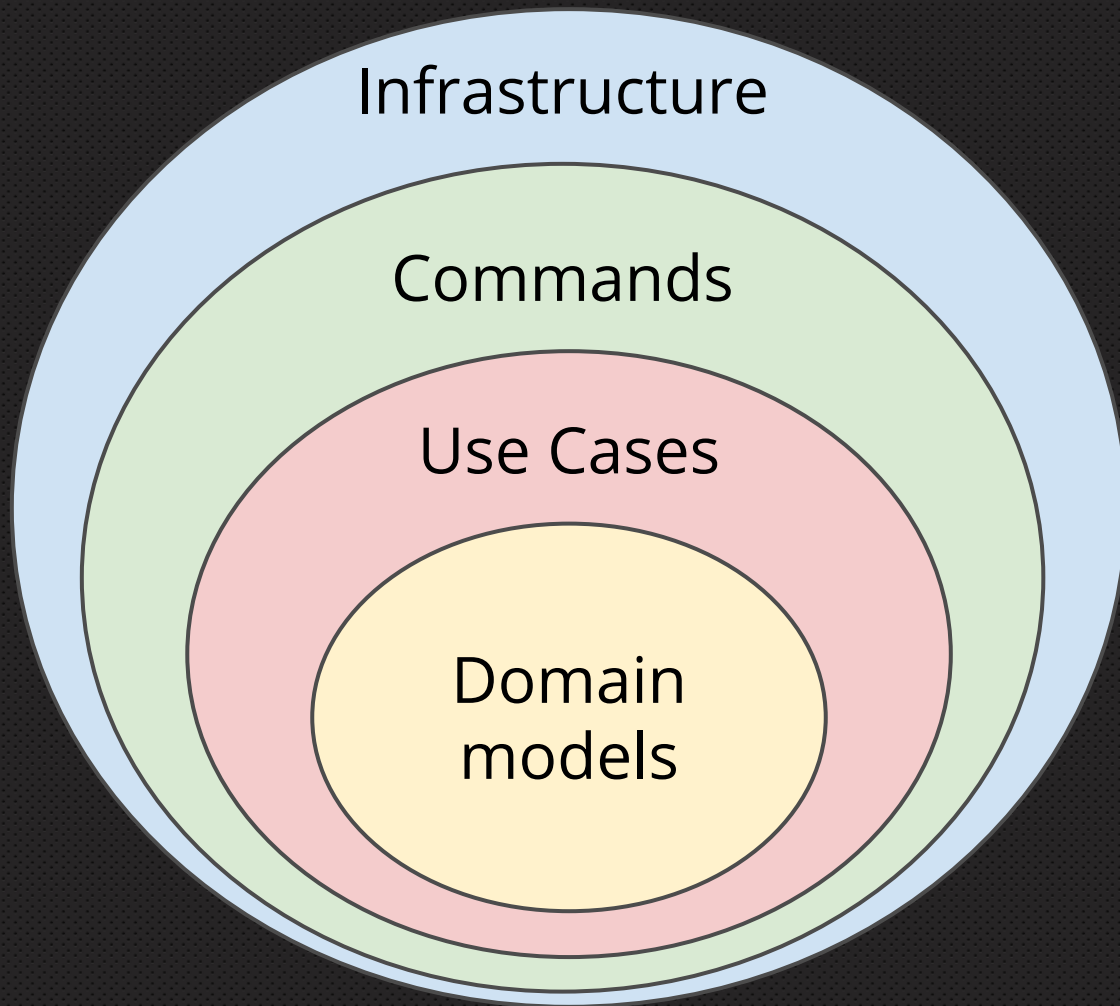
Your Domain First



Domain
models







Clean Code!

Clean code

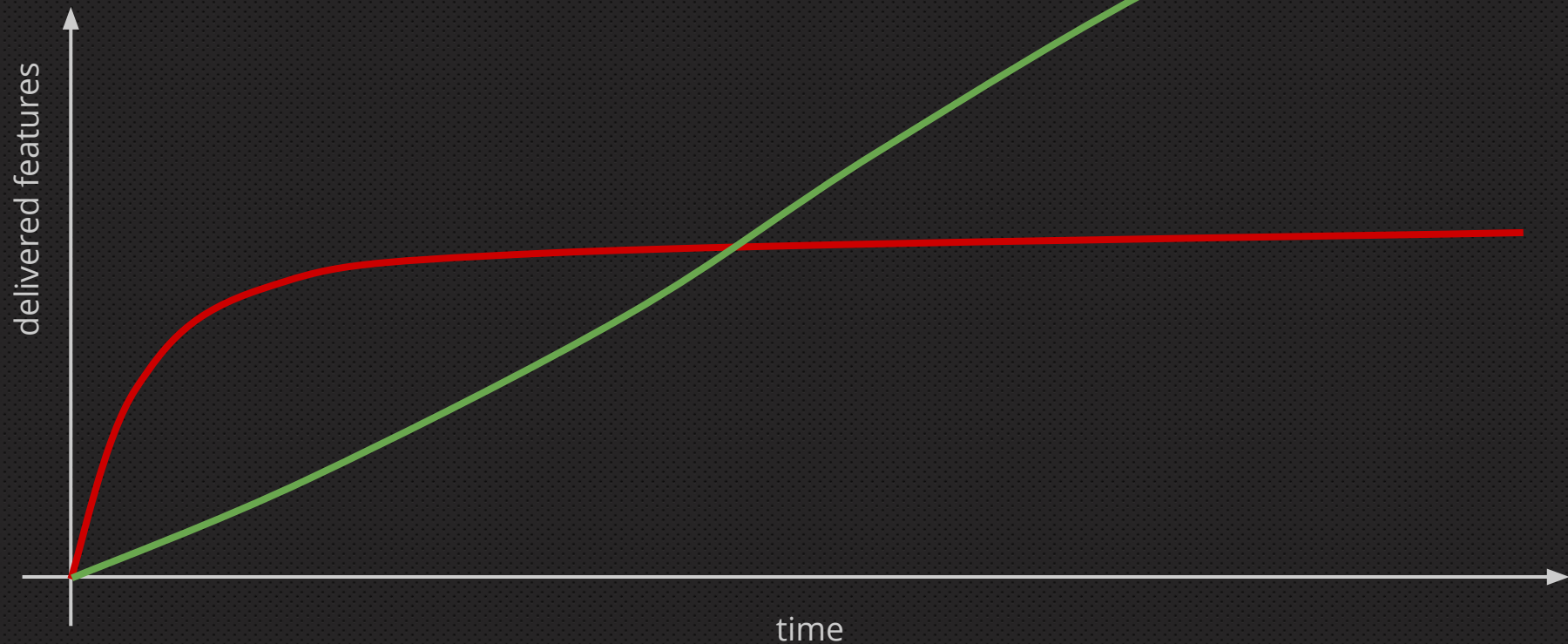
- decoupled
- reusable
- testable
- maintainable
- changeable

Sustainable!

Adding new features?

Easy!

Features delivered over time





Recap

It's your choice

Start the easy way

and make the changes **hard**

Start the hard way

and make the changes **easy**



*“Everything is hard
before it is easy”*

- Goethe



Questions?

Thank you!



https://speakerdeck.com/super_marek

@super_marek