Вагиф Абилов

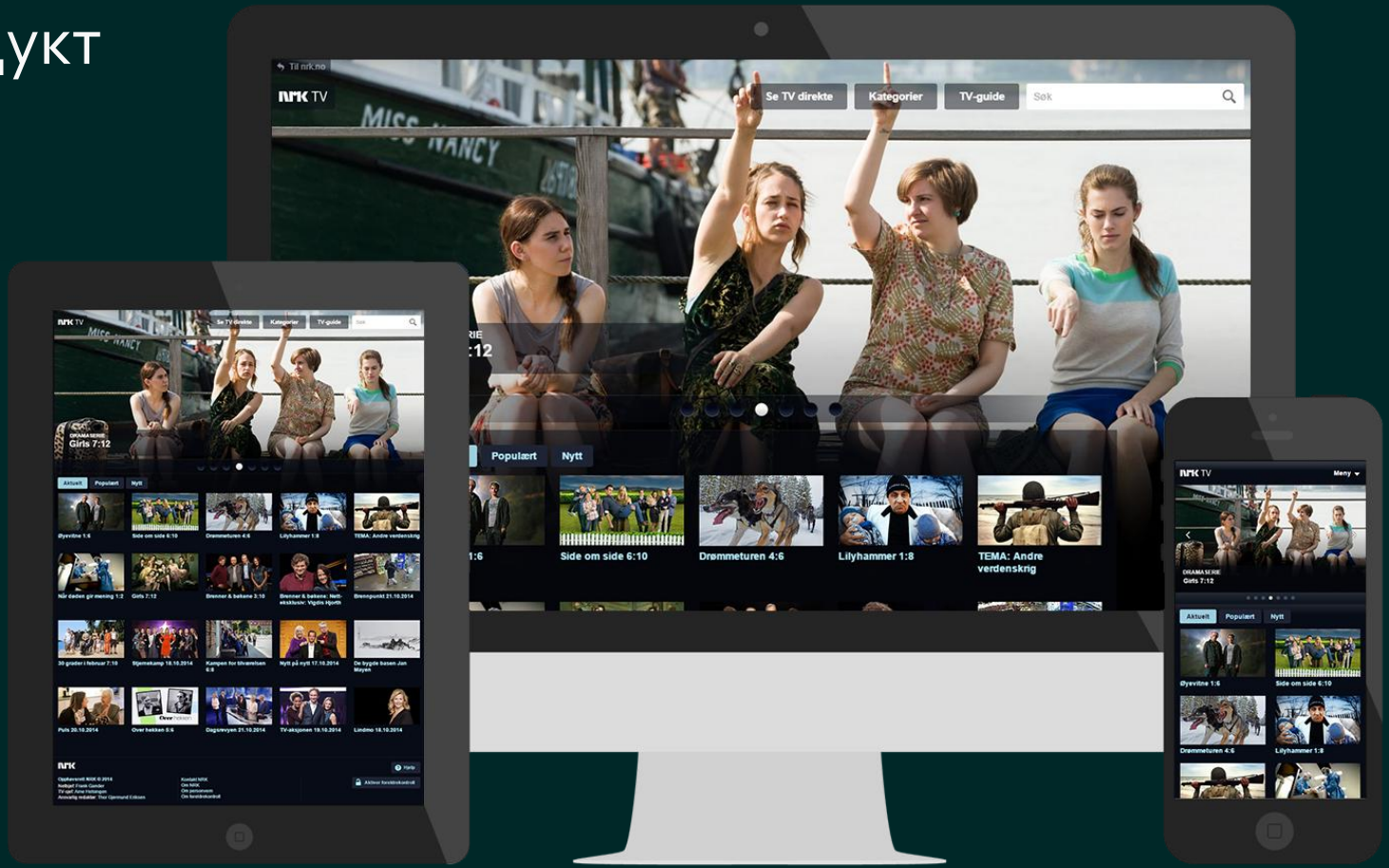
Три года с F# в продакшн
Можно ли это назвать успехом?

Вагиф Абилов
Консультант в Miles
Россия - Норвегия

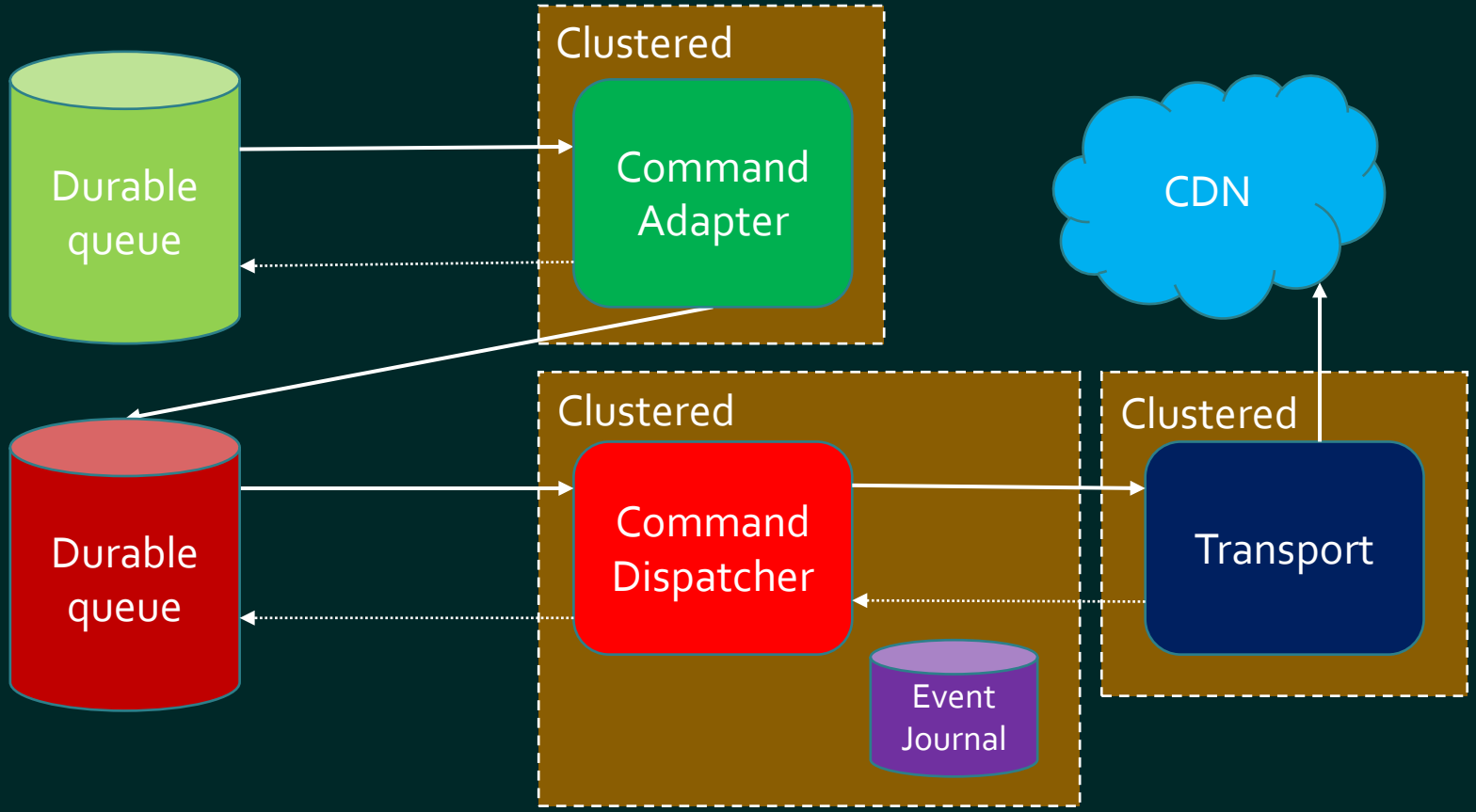
Работаю с F# и C#

@ooobject
vagif.abilov@mail.com

Наш продукт


Se TV direkte
Kategorier
TV-guide
Søk
Populært
Nytt
Side om side 6:10
Drømmeturen 4:6
Lilyhammer 1:8
TEMA: Andre verdenskrig
Aktuelt
Meny

Наша архитектура



Наши инструменты

- F#, модель акторов и Akka.NET
- Библиотека Akkling (Akka.NET F# API)
- Провайдеры типов F# для JSON, Yaml и SQL
- Microsoft SQL Server для хранения журнал персистентых акторов
- Очереди RabbitMQ
- Web API на основе Suave.IO
- Тестирование с использованием XUnit, FsCheck и TickSpec

Из комментариев к интервью на Хабре

Я правильно понял, что на данный момент из реально завершенных «фунциональных» проектов у автора только «Игра жизни Конвея» размером в 14 строк?

Комментарий к беседе F# (тоже на Хабре)

Если F# так хорош, как вы его описываете, почему его никто не хочет знать?

Наиболее желанные языки (StackOverflow, опрос 2018)

Python	25.1%
JavaScript	19%
Go	16.2%
Kotlin	12.4%
TypeScript	11.9%
Java	10.5%
C++	10.2%
Rust	8.3%

C#	8.0%
Swift	7.7%
…	…
CSS	7.6%
SQL	6.8%
…	…
F#	4.0%
…	…

Наиболее оплачиваемые технологии (там же)

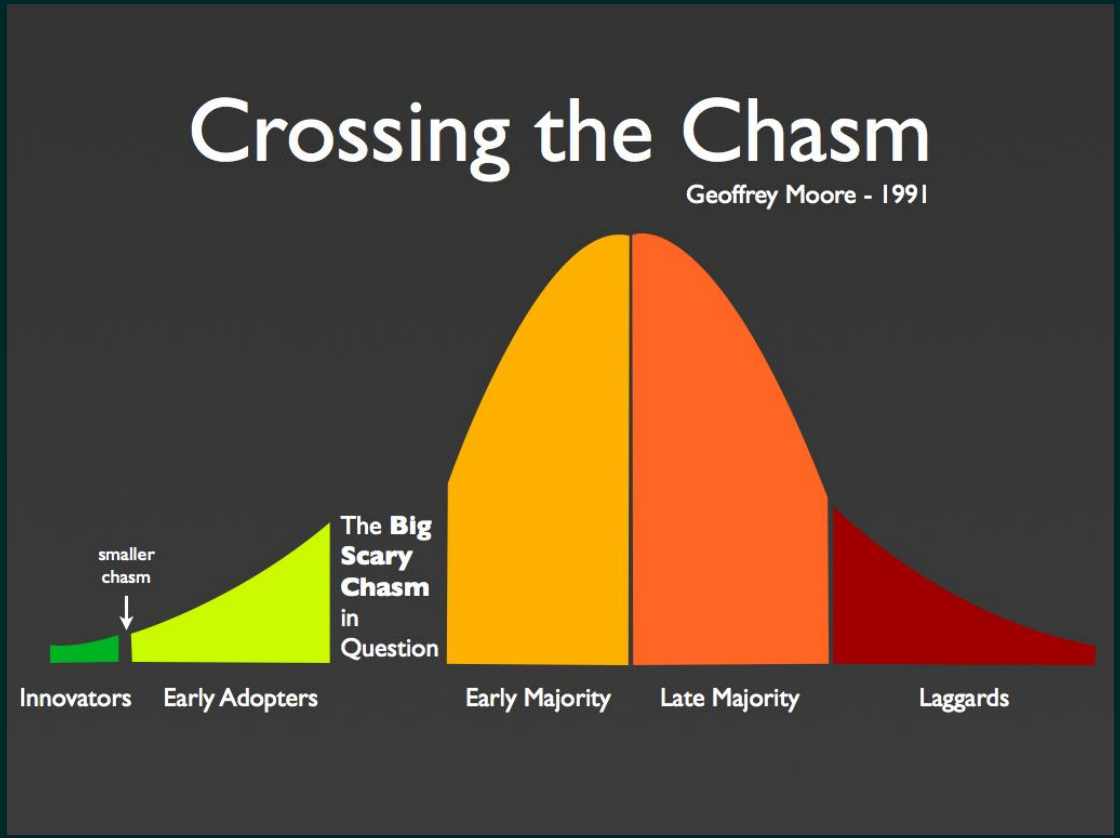
F#	$74,000
Ocaml	$73,000
Clojure	$72,000
Perl	$69,000
Rust	$69,000
Erlang	$67,000
Scala	$67,000
Go	$66,000

Ruby	$64,000
Bash/Shell	$64,000
Coffee Script	$63,000
Haskel	$60,000
Julia	$60,000
TypeScript	$60,000
C#	$59,000
Objective-C	$58,000

Этот доклад – не продолжение
тлеющей войны за
единственно верный
язык программирования

Речь пойдет о наболевшем у прагматиков
(“pragmatists in pain”,
по выражению Эрика Синка)

Цикл адаптации технологии по Джеффри Муру



Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~

Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~


Thread Lock State

Immutable collections .NET

Under the hood

Dmitry Ivanov, JetBrains

```
class Point {

int X { get; set; }
int Y { get; set; }

Point(int x, int y) { X = x; Y = y }

void IncreaseX (int xOffset) { X += xOffset; }
void IncreaseY (int yOffset) { Y += yOffset; }

}
```

```
class Point {

int X { get; set; }
int Y { get; set; }

Point(int x, int y) { X = x; Y = y }

void IncreaseX (int xOffset) { X += xOffset; }
void IncreaseY (int yOffset) { Y += yOffset; }

int GetHashCode() {…}
bool Equals(object other) {…}
}
```

```
class Point {

readonly int X;
readonly int Y;

Point(int x, int y) { X = x; Y = y }

Point IncreaseX (int xOffset) => new Point(x + xOffset, y);
Point IncreaseY (int xOffset) => new Point(x, y + yOffset);

int GetHashCode() {…}
bool Equals(object other) {…}
}
```

Это мы еще не начали
с коллекциями разбираться...

Последствия
~~неверного шага~~
недостаточного опыта

Принципиальная разница в выборе начальных установок

Закон Амдала в действии

Если у вас 10 процессоров,
но вы распараллеливаете лишь 40% кода,
то быстродействие увеличивается в 1,56 раза

Но сложности возникают не только с параллелизацией – трудно понять, что происходит внутри мутабельных структур

```
class Point {

int X { get; set; }
int Y { get; set; }

Point(int x, int y) { X = x; Y = y }

void IncreaseX (int xOffset) { X += xOffset; }
void IncreaseY (int yOffset) { Y += yOffset; }

}
```

Демо
Структуры данных на F#

Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~

Раз уж мы упомянули «Игру жизни» Конвея

Релизация «Игры жизни» Конвея

```
let isAlive population cell =
    population
    |> List.exists ((=) cell)

let aliveNeighbours population cell =
    neighbours cell
    |> List.filter (isAlive
population)

let survives population cell =
    aliveNeighbours population cell
    |> List.length
    |> fun x -> x >= 2 && x <= 3

let reproducible population cell =
    aliveNeighbours population cell
    |> List.length = 3
```

```
let allDeadNeighbours population =
    population
    |> List.collect neighbours
    |> Set.ofList |> Set.toList
    |> List.filter (not << isAlive
population)

let nextGeneration population =
    List.append
        (population
        |> List.filter
            (survives population))
        (allDeadNeighbours population
        |> List.filter
            (reproducible population))
```

Разрешение конкретного типа

```
let neighbours (x, y) =
    [ for i in x-1..x+1 do
      for j in y-1..y+1 do
      if not (i = x && j = y) then yield (i,j) ]

let neighbours (x, y, z) =
    [ for i in x-1..x+1 do
      for j in y-1..y+1 do
      for k in z-1..z+1 do
      if not (i = x && j = y && k = z) then yield (i,j,k) ]
```

Демо
Вывод типов в F#

Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~

Демо
Алгебраические типы в F#

Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~

Из мудрых мыслей Твиттера

«Программировать на Java — все равно что заниматься русской литературой: вам нужно определить сотню имен, прежде чем начнут происходить какие-то события»

@jamesiry

Бизнес-объекты

Бизнес-объекты

Граждане первого сорта

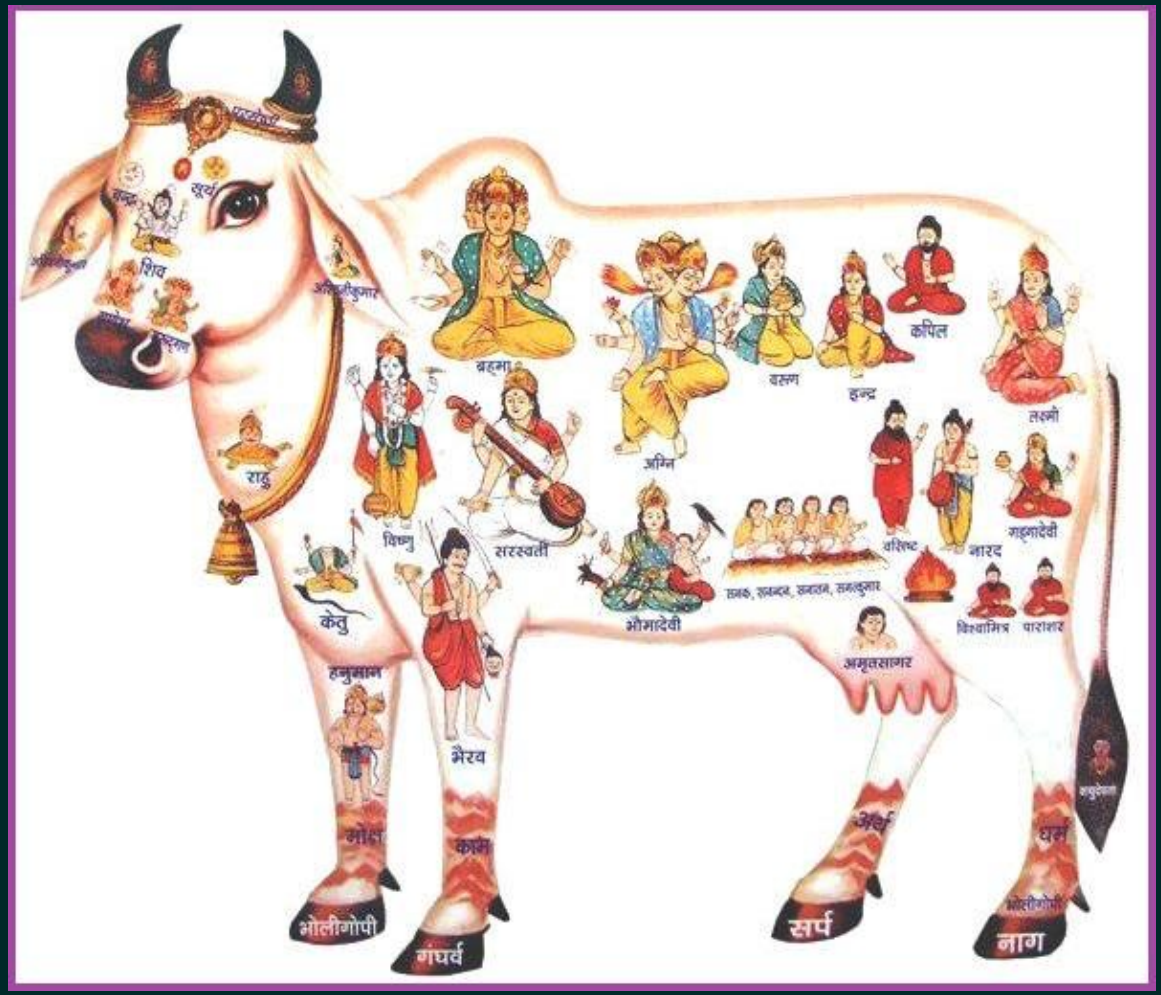
DTO

DTO

Нелегальные мигранты

Local DTO

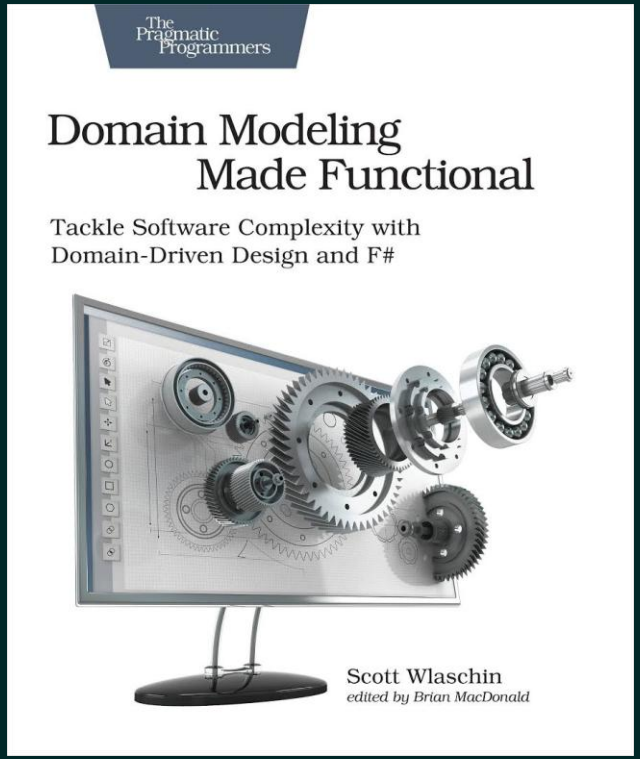
Легальные мигранты


शिव
सूर्य
ब्रह्मा
अग्नि
वरुण
इन्द्र
कपिल
लक्ष्मी
राहु
विष्णु
सरस्वती
केतु
भौमादेवी
सनक, सनन्दन, सनातन, सनत्कुमार
वशिष्ठ
नारद
गङ्गादेवी
विश्वामित्र
पाराशर
अमृतसागर
हनुमान
भैरव
मोक्ष
काम
अर्थ
धर्म
भोलीगोपी
गंधर्व
सर्प
नाग

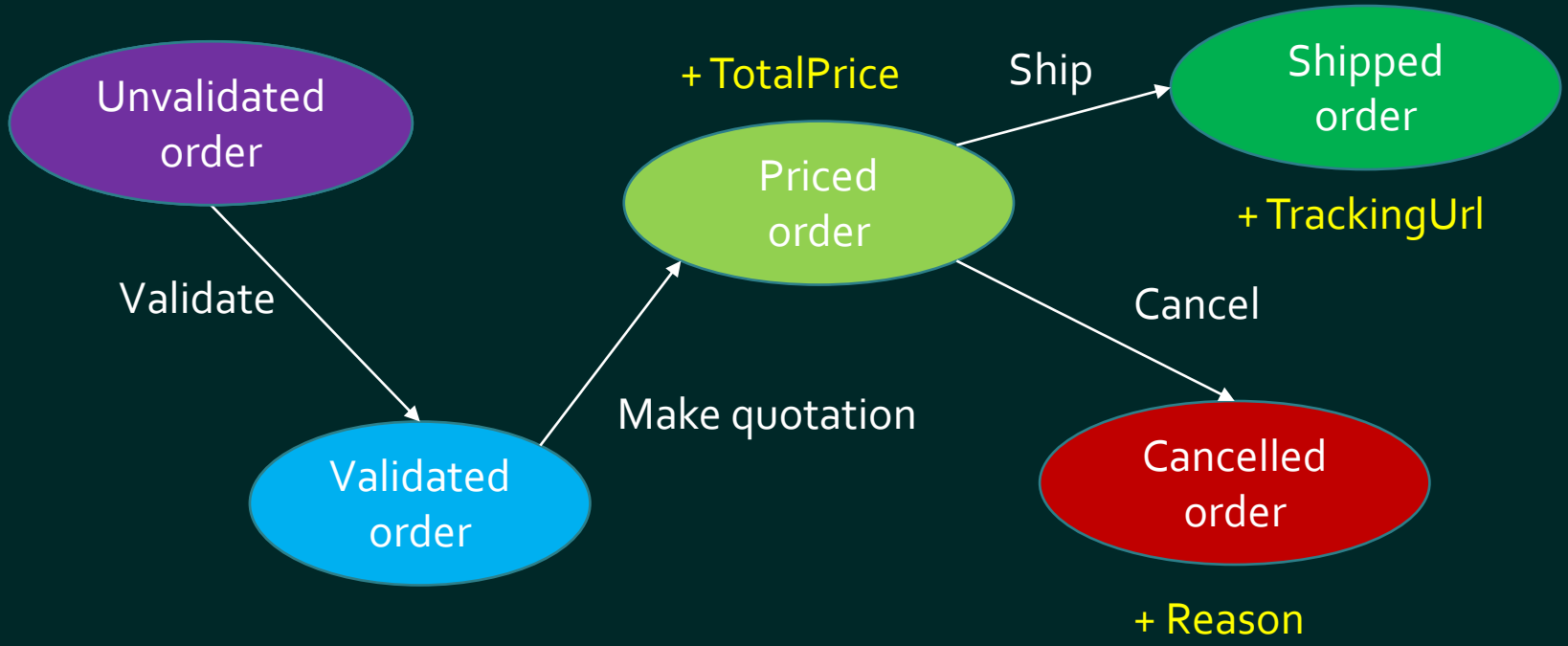
Мартин Фаулер о локальных DTO

«Один из сценариев, когда стоит пользоваться чем-то типа DTO – это когда имеется существенное несовпадение модели слоя презентации и доменной модели нижнего уровня.»

Скотт Влащин “Функциональное доменное моделирование”



Оформление заказа



```
class Order {
…
decimal TotalPrice { get; }
Uri TrackingUrl { get; }
string CancellationReason { get; }


bool IsValidated { get; }
bool IsShipped { get; }
bool IsCancelled { get; }
}
```

```
class Order {
…
decimal TotalPrice { get; }
Uri TrackingUrl { get; }
string CancellationReason { get; }

bool IsValidated { get; }
bool IsShipped { get; }
bool IsCancelled { get; }

void Validate();
void Ship();
void Cancel();
}
```

```
class Order {
…
decimal TotalPrice { get; }
Uri TrackingUrl { get; }
string CancellationReason { get; }

bool IsValidated { get; }
bool IsShipped { get; }
bool IsCancelled { get; }
}

class OrderManager {
void Validate(Order order);
void Ship(Order order);
void Cancel(Order order);
}
```

```
class Order {
…
decimal TotalPrice { get; }
Uri TrackingUrl { get; }
string CancellationReason { get; }

bool IsValidated { get; }
bool IsShipped { get; }
bool IsCancelled { get; }
}

class OrderManager {
void Validate(Order order);
void Ship(Order order);
void Cancel(Order order);
}
```

Чистые объекты

Чисто бизнес ничего личного

```
class UnvalidatedOrder { … }

class ValidatedOrder { … }

class PricedOrder {
… decimal TotalPrice { get; }
}

class ShippedOrder {
… Uri TrackingUrl { get; }
}

class CancelledOrder {
… string Reason { get; }
}
```

```
class OrderValidator {
  ValidatedOrder
    ValidateOrder(…)
}

class QuotationMaker {
  PricedOrder
    MakeQuotation(…)
}

class OrderDispatcher {
  ShippedOrder
    ShipOrder(…)
}
```

Демо
Доменное моделирование в F#

Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~

Существенным при отказе от null
является не переход на option,
а по возможности отказ и от option

Ярон Мински

«Сделать незаконное состояние непредставляемым»
(Make illegal state unrepresentable)

https://blog.janestreet.com/effective-ml-revisited/

Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~

Статистика проекта энергосектора (Саймон Казинс)

Implementation	C#	F#
Braces	56,929	643
Blanks	29,080	3,630
Null Checks	3,011	15
Comments	53,270	487
Useful Code	163,276	16,667
App Code	305,566	21,442
Test Code	42,864	9,359
Total Code	348,430	30,801

Метрики первой версии нашего проекта

Media Distribution Engine	C#	F#
Files	195	56
Total lines	12089	6042
Empty lines	1325	913
Comment lines	242	32
Braces lines	3398	57
Namespace lines	1233	525
Code lines	5891 (55%)	4515 (88%)

Лечение наболевшего

1. Неизменяемость структур данных (immutability)
2. Вывод типов (type inference)
3. Алгебраические типы и распознавание шаблонов (pattern matching)
4. Ненужность бизнес-объектов
5. Отсутствие null
6. Компактность кода
7. ~~Мощные библиотеки коллекций~~
8. ~~Провайдеры типов~~
9. ~~Тестирование на основе свойств (property-based testing)~~

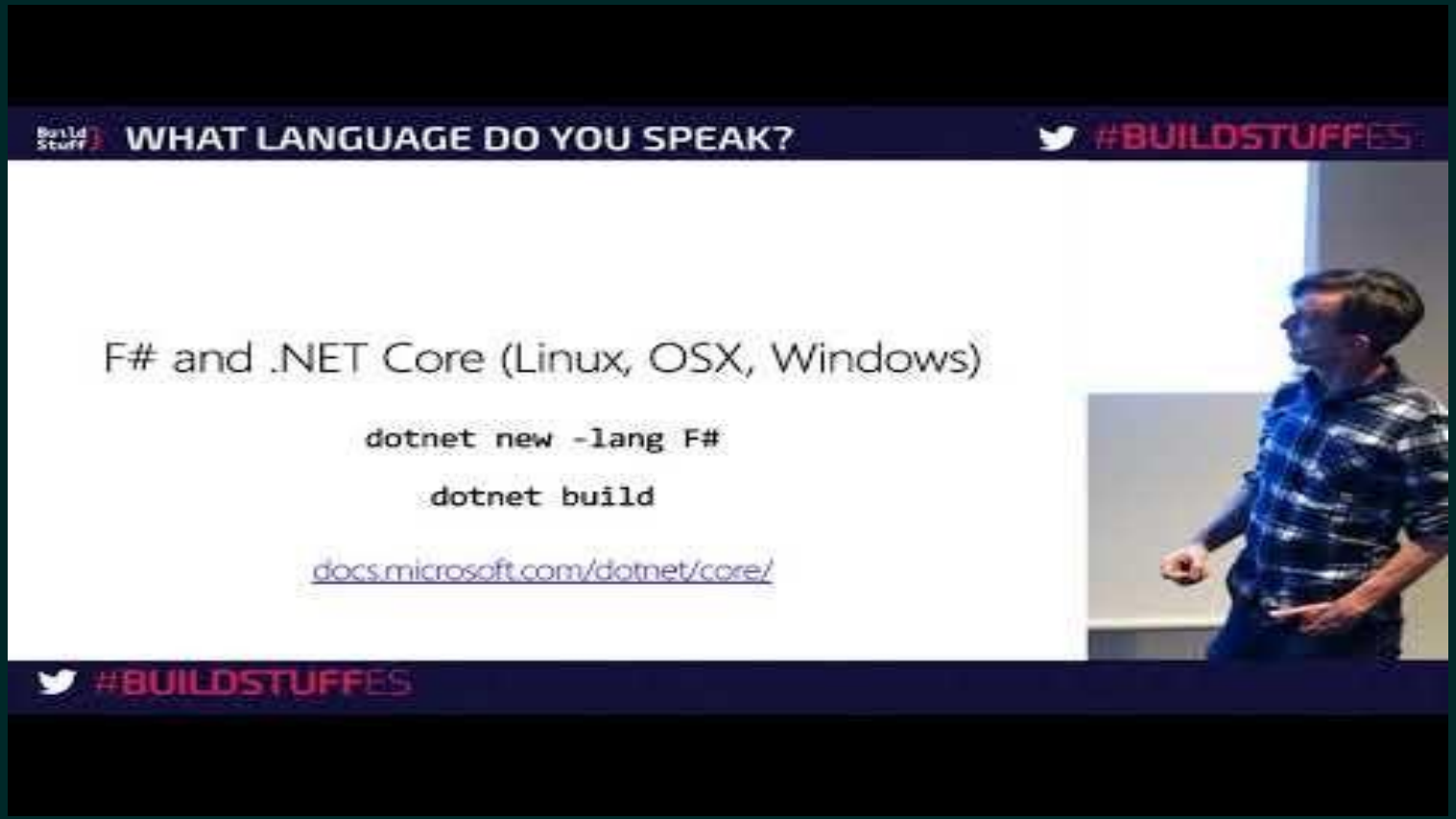
Сдали бы мы наш проект, реализовав его на C#?

Безусловно!
Но...

Какое существенное преимущество нам дал выбор F#?

Значительно сократил
цикл реализации
функциональных требований

Дон Сайм - F#, который я люблю



Как НЕ начать писать на понравившемся языке?

1. Следить за рейтингами популярности технологий
2. Опасаться сложностей поиска проекта
3. Убеждать себя, что лучшее – враг хорошего
4. Ждать, пока возникнет возможность на текущей работе

Как начать писать на понравившемся языке?

Садитесь
и
пишите!

Спасибо!

Вагиф Абилов
Консультант в Miles
Россия - Норвегия

@ooobject
vagif.abilov@mail.com