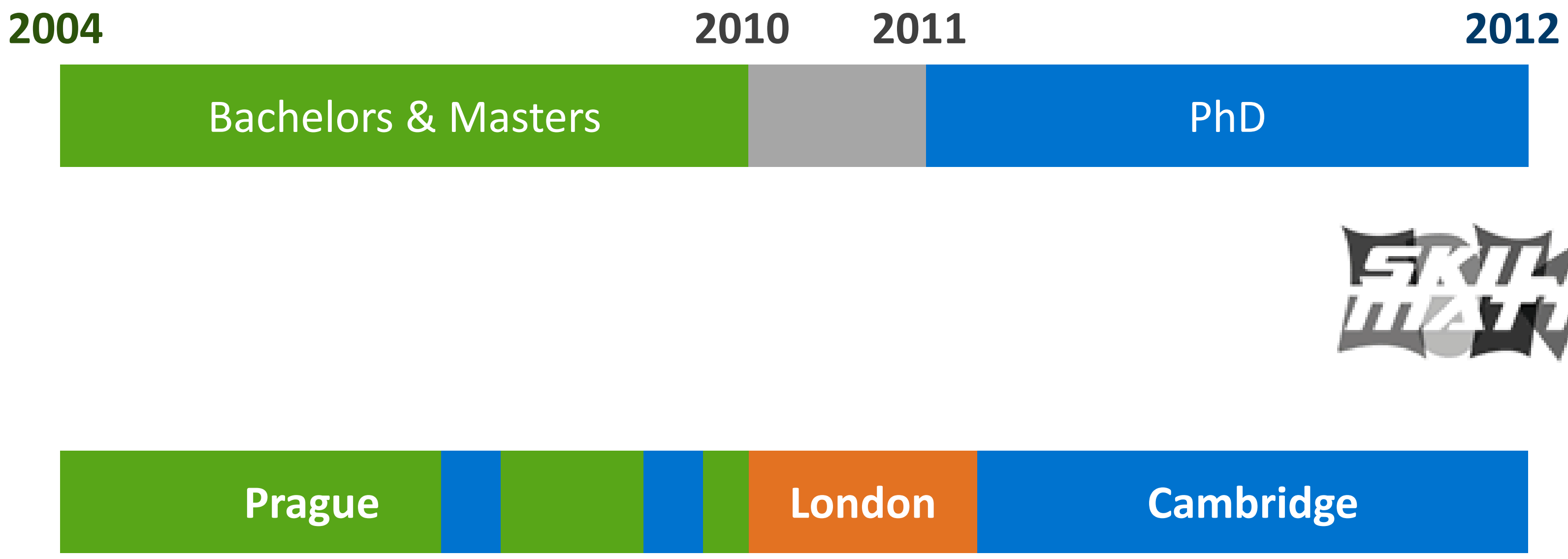
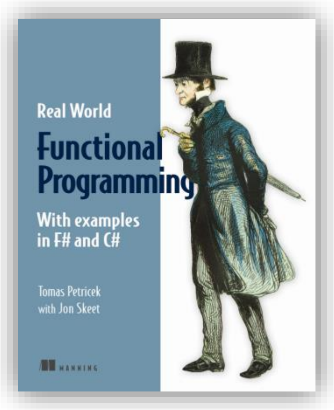
# Teaching Functional Programming to **Professional .NET Developers**

**Tomas Petricek**
University of Cambridge

# About me and my background

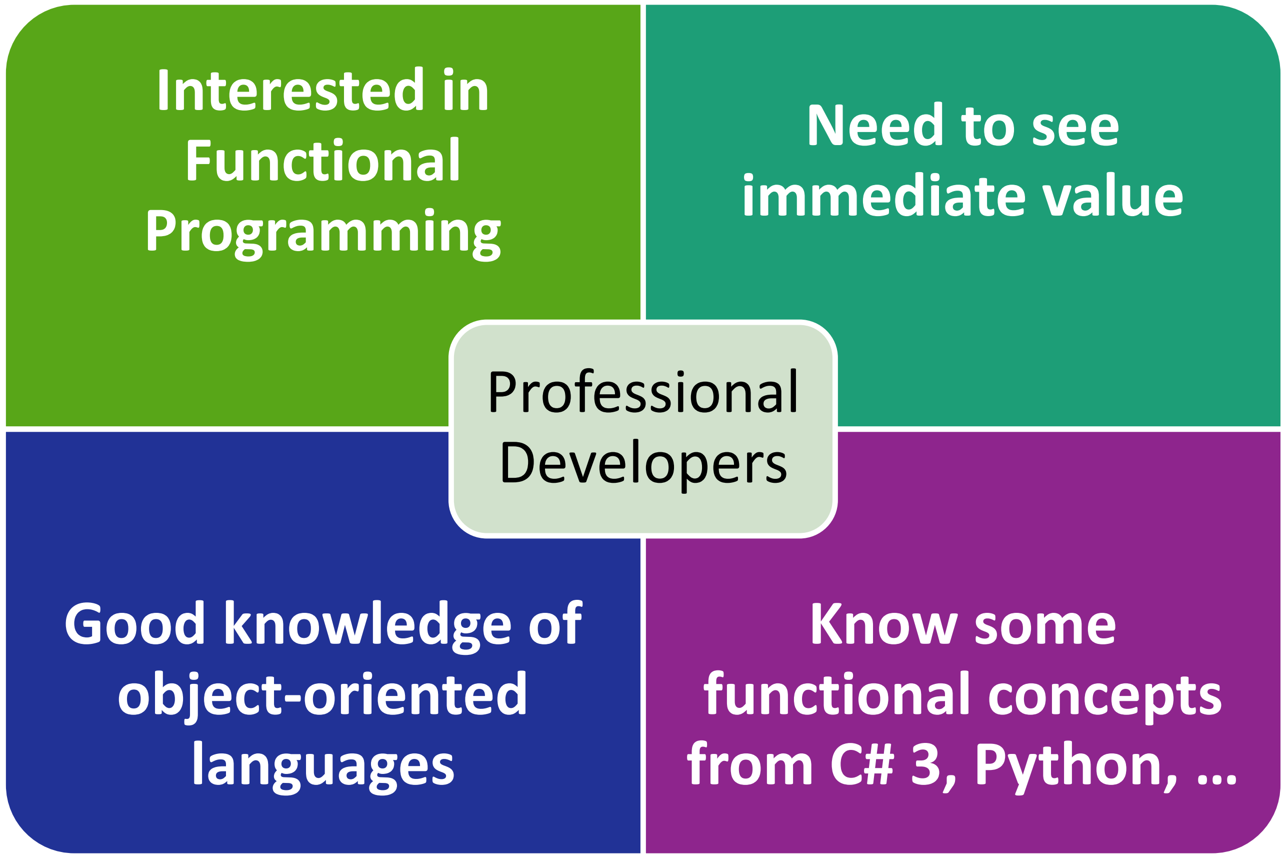




Functional programming for

**Professional .NET Developers**

# Different target audience


Interested in Functional Programming
Need to see immediate value
Professional Developers
Good knowledge of object-oriented languages
Know some functional concepts from C# 3, Python, …

# Challenges and our approach

**Conveying new concepts**

- Start with familiar language features
- Demonstrate concepts using C#

**Relation with object-oriented**

- Functional types and domain models
- Relations with OO design patterns

**Benefit from FP in the industry**

- Think about problems differently
- Many concepts can be used in C# too

## **Demonstration #1:** Immutability and fluent interfaces

# Fluent interface pattern

## Simplify object construction

```
var tea = Product.Create("Earl Gray Tea")
                 .WithPrice(10.0M)
                 .WithPromotion();
```

## Creates and mutates an object

```
public Product WithPrice(string price) {
  this.price = price;
  return this;
}
```

# Fluent interface pattern

## Avoiding code duplication

```
var tea1 = Product.Create("Earl Gray Tea")
                  .WithPrice(10.0M);
var tea2 = Product.Create("Earl Gray Tea")
                  .WithPrice(12.0M);
```

## Does this behave the same?

```
var tea = Product.Create("Earl Gray Tea");
var tea1 = tea.WithPrice(10.0M);
var tea2 = tea.WithPrice(12.0M);
```

## Fluent interface pattern

Fixed using **immutable types**

- Easy to change in **C#**
- Even easier using **F#** records

Example **summary**

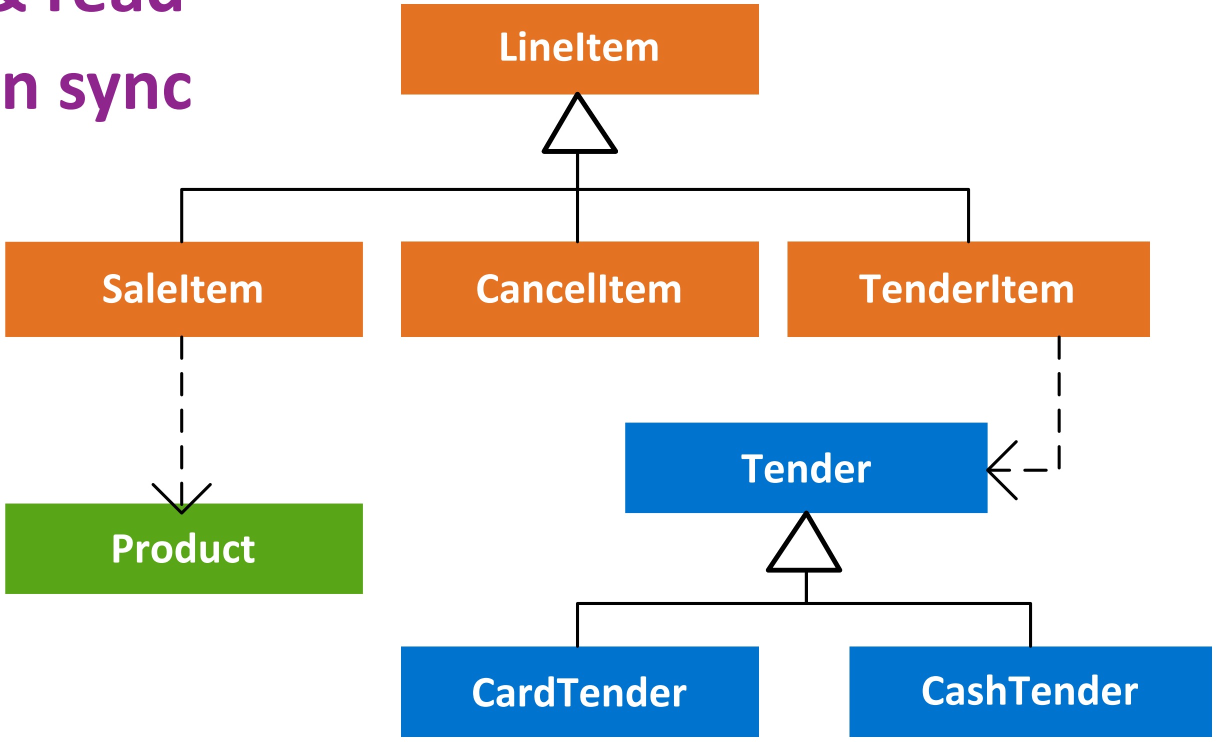
- Show problem in a **familiar setting**
- Immutability leads to **correct code**
- **More important** than parallelism

## **Demonstration #2:** Functional types and domain modeling

# Modeling the problem domain

Modeling using **UML diagrams**

- Capture the **idea**
- Easy to **draw & read**
- Hard to **keep in sync**



# Modeling the problem domain

F# types fit on a **single slide**

```
type Price = decimal
type Code = string
type Quantity = int
type Product = string * Code * Price
```

```
type Tender =
   | CashTender
   | CardTender of string
```

```
type LineItem =
   | SaleItem of int * Product * Quantity
   | TenderItem of Tender * Price
   | CancelItem of int
```

# Modeling the problem domain

**Modeling domain** using F#

- Simple **declarative specification**
- Teaches how **F# types** are compiled
- Focus on **data** rather than **operations**

**Functional types** in practice

- May be used for **prototyping**
- Ideally part of the **codebase**
- Easy **integration** is crucial

## Conclusions

## Read the paper for more!

There is a way to use **existing knowledge!**

- Implement **functional concepts** in C# or Java
- Show how FP relates to **common patterns**
- Give **takeaways** usable in any language

More information

- Real-World FP book: http://manning.com/petricek
- FP and F# Trainings: http://skillsmatter.com
- Contact & more: http://tomasp.net