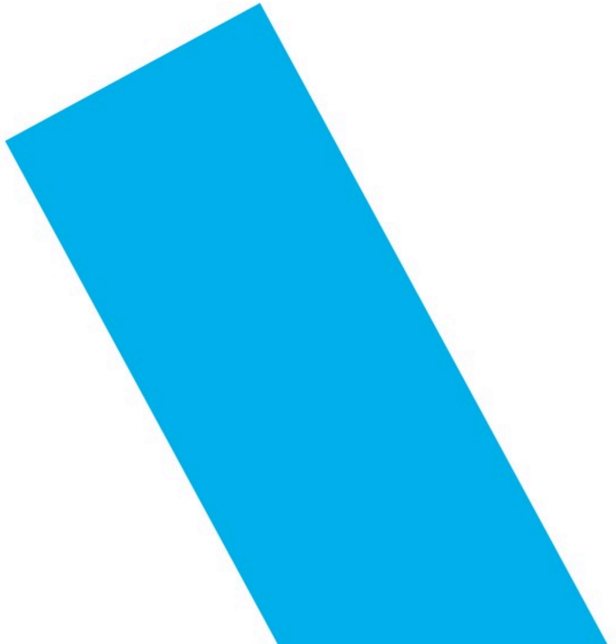
Going Functional with

Railway Oriented Programming in C#

.NET Stammtisch Linz
May 27th, 2025



Who am I?

Paul Rohorzka

Principal Software Engineer

Software Gardener @SQUER

SQUER

Who am I?

Paul Rohorzka

Principal Software Engineer

Software Gardener @SQUER

Created in cooperation with

David Walser

Code Composter @SQUER

SQUER

What to Expect

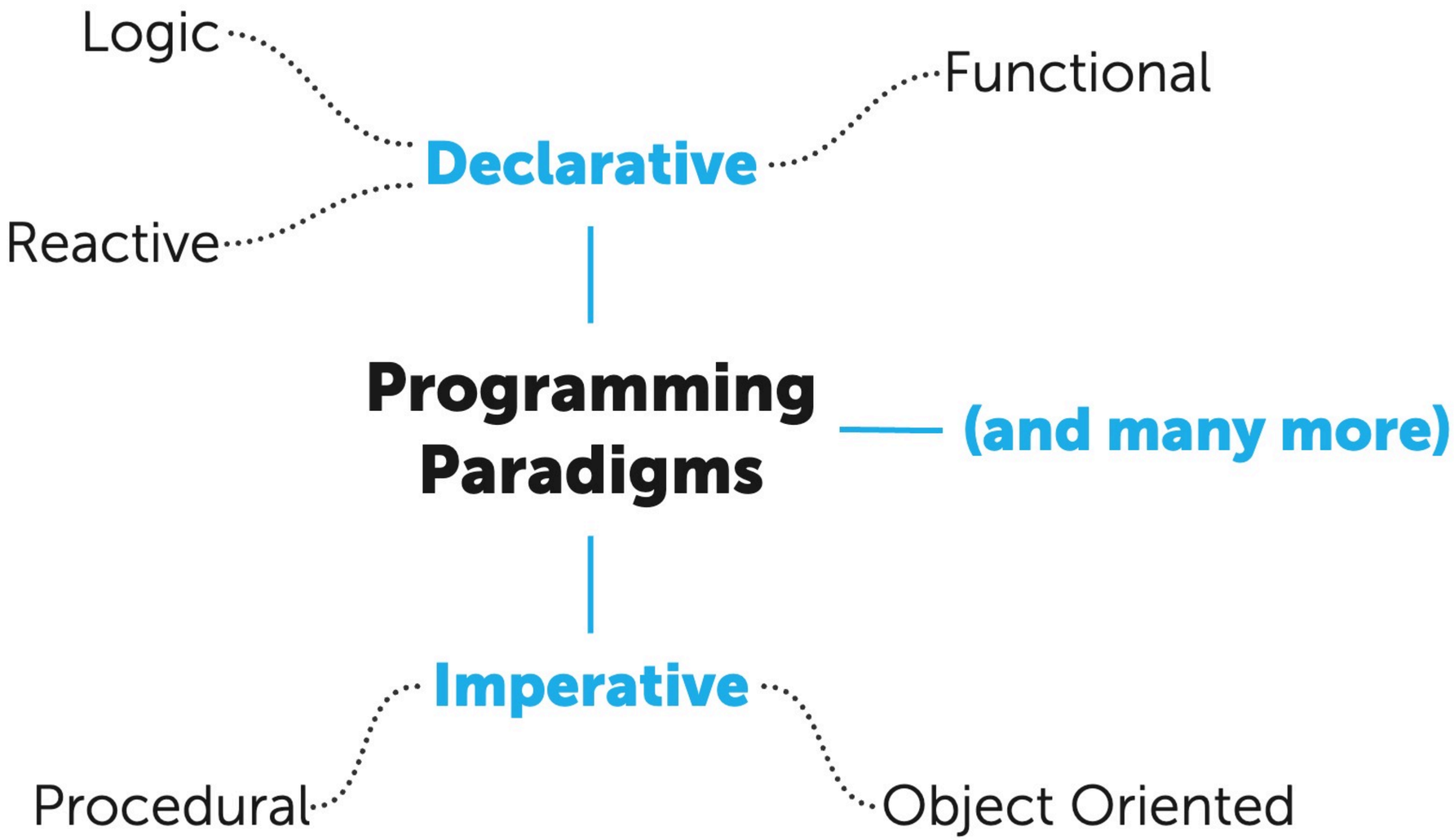
Programming Paradigms

Railway Oriented Programming

Functional in C#

Real-World Experiences

SQUER


Logic
Functional
Declarative
Reactive
Programming Paradigms
(and many more)
Imperative
Procedural
Object Oriented

SQUER

Imperative

```
var sum = 0;
foreach (var number in numbers)
{
    if (number % 2 == 0)
    {
        sum += number;
    }
}
return sum;
```

Declarative

```
return numbers
        .Where(n => n % 2 == 0)
        .Sum();
```

SQUER

Functional, yes?

So...

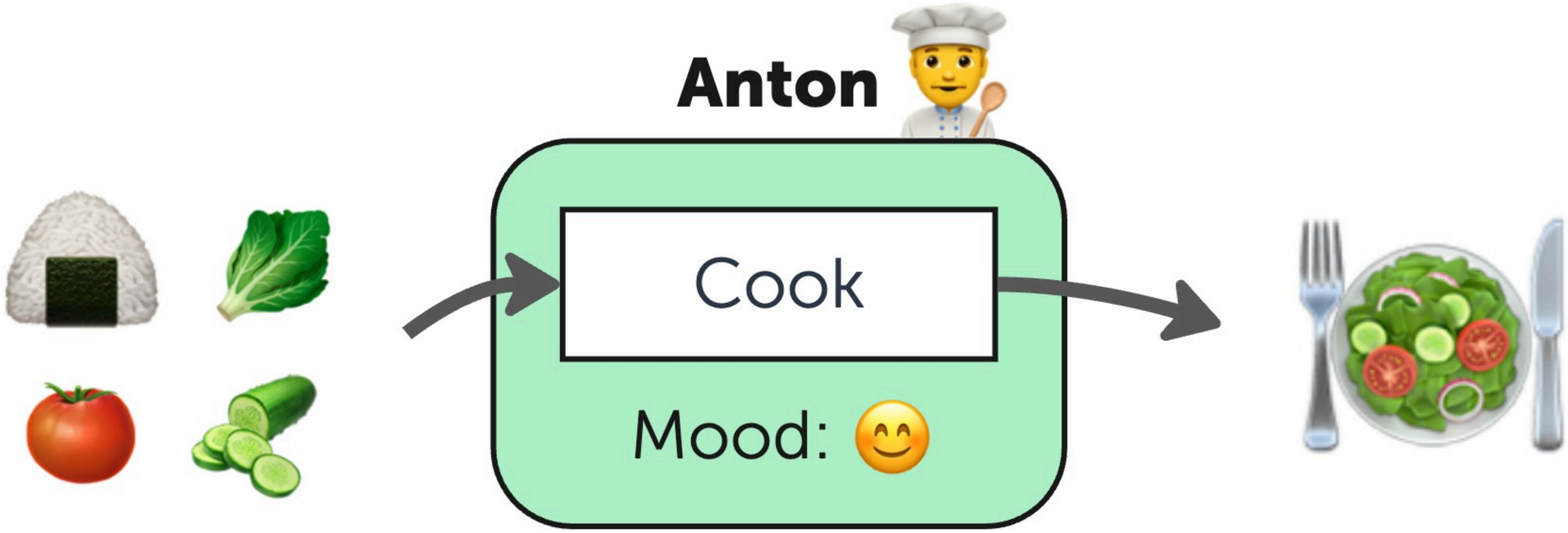
⚠ **Functions!** ⚠

SQUER

Let's Cook!

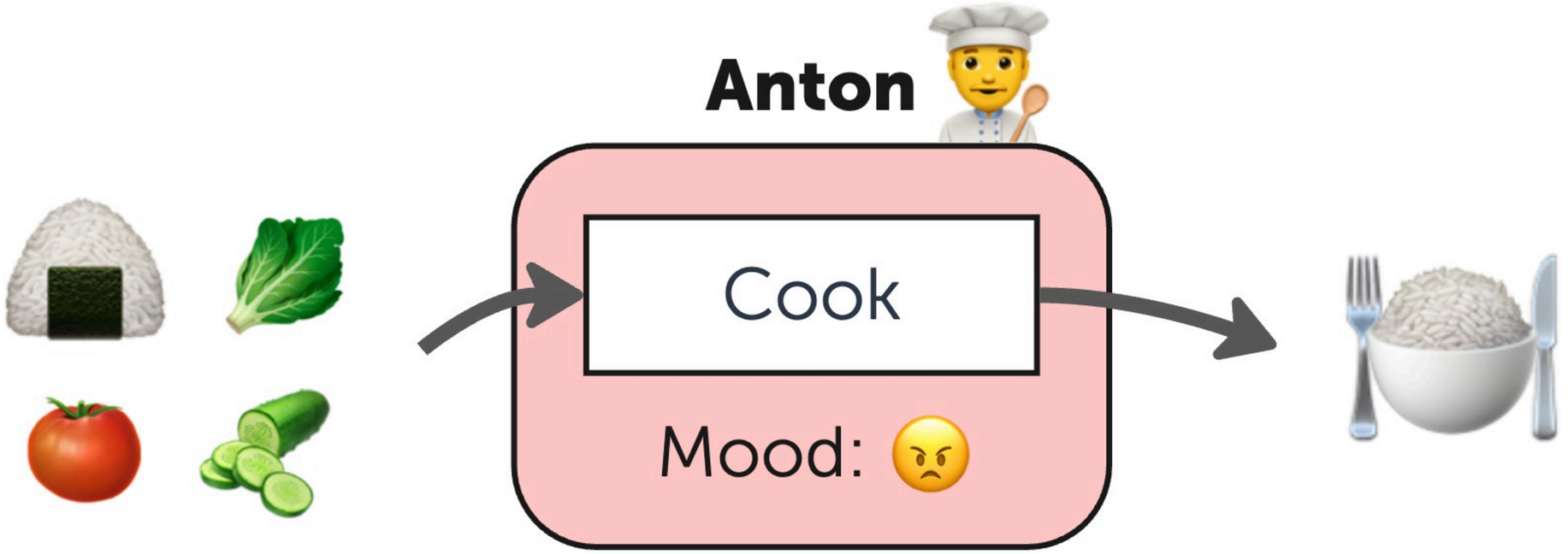
SQUER

A Happy Chef



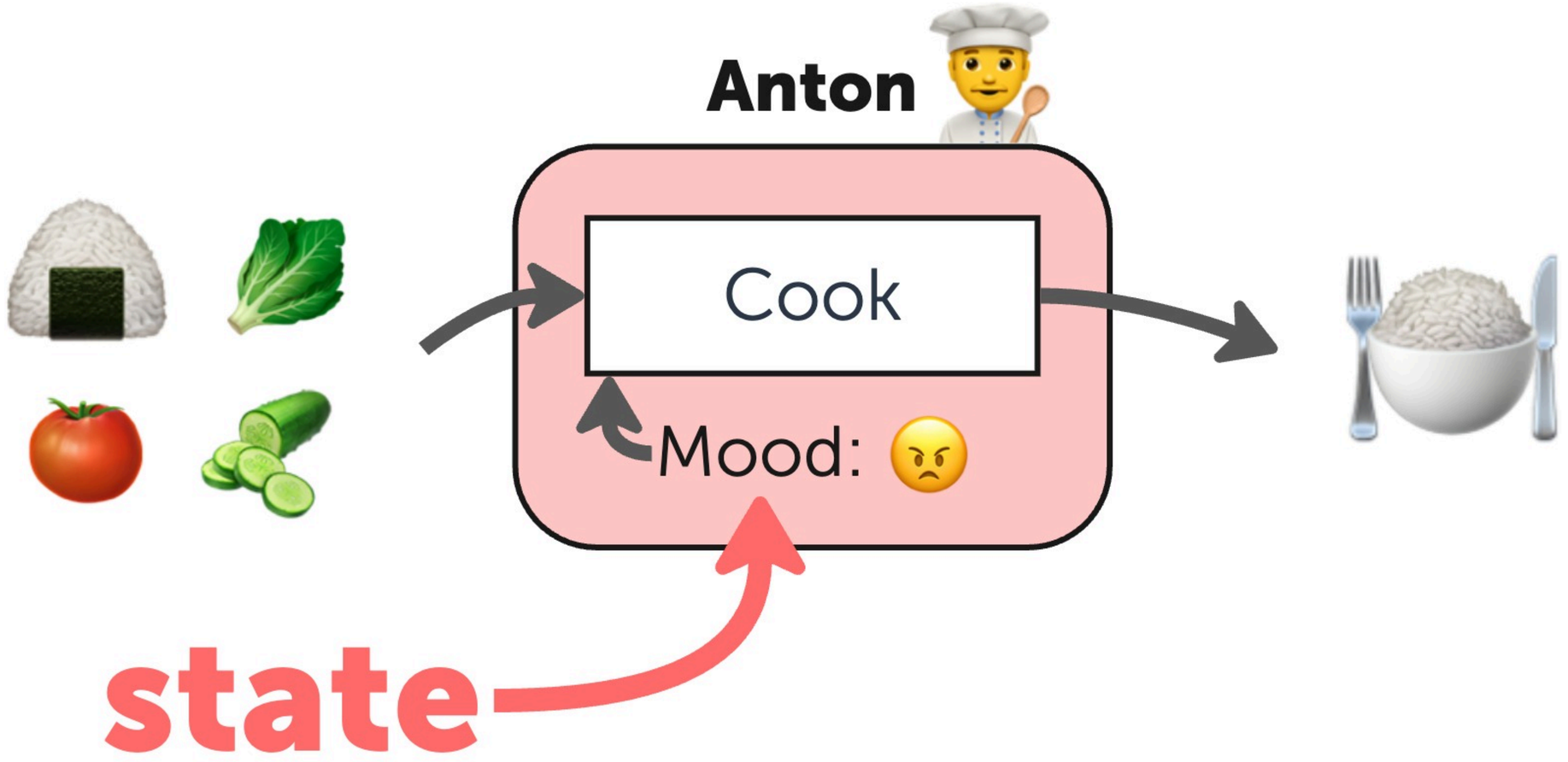
SQUER

A Grumpy Chef



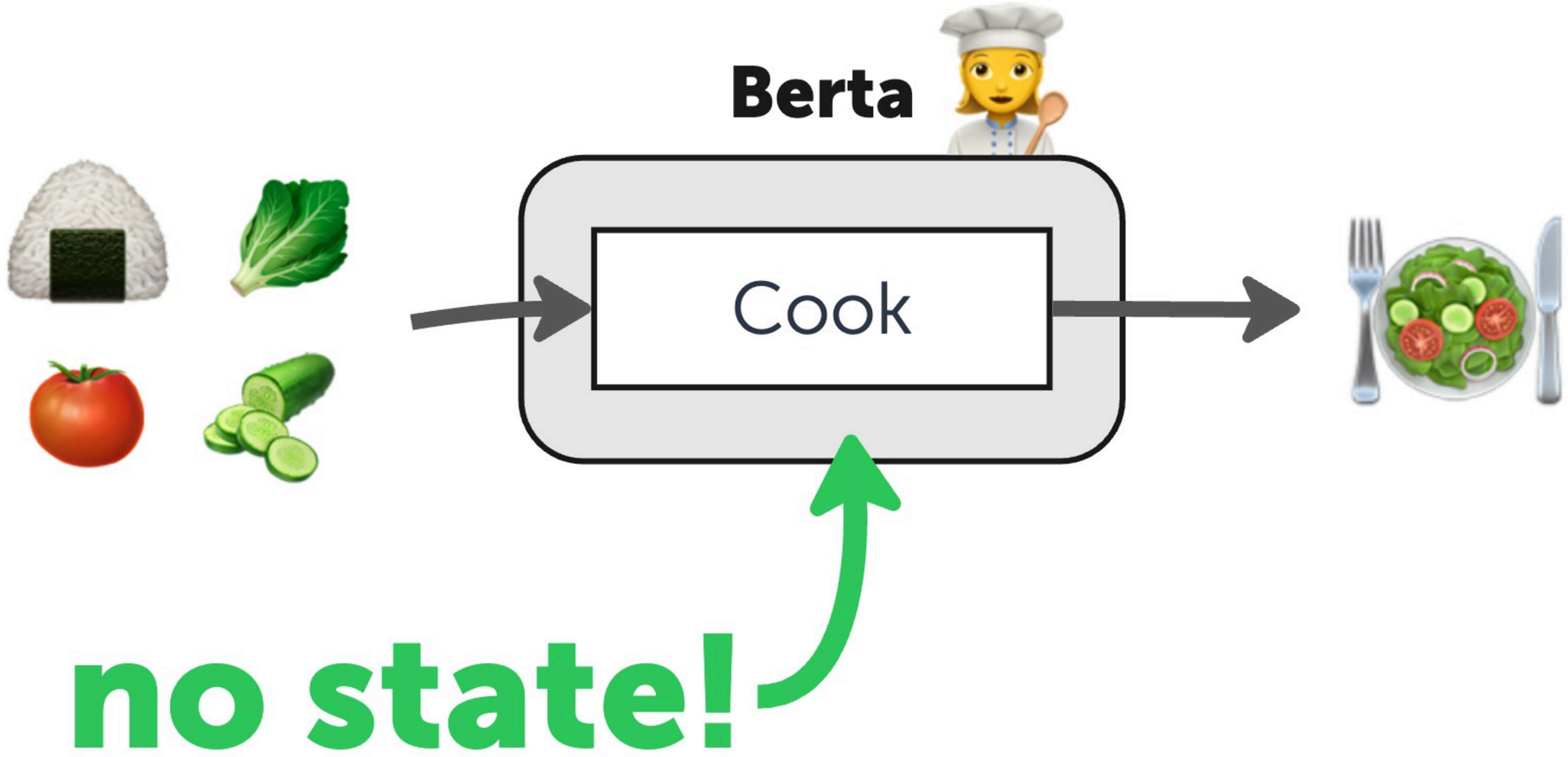
SQUER

A Stateful ~~Chef~~ Method



SQUER

A Stateless ~~Chef~~ *Method*



SQUER

Pure Function

```
int Add(int x, int y)
{
    return x + y;
}
```

```
var result = Add(5, 2); //7
```

SQUER

Pure Function

```
int Add(int x, int y)
{
    return x + y;
}
```


```
var result = Add(5, 2); //7
```

SQUER

Pure Function

```
int Add(int x, int y)
{
    return x + y;
}
```

~~Side effects~~

```
var result = Add(5, 2); //7
```

SQUER

Pure Function

```
int Add(int x, int y)
{
    return x + y;
}
```

Deterministic

~~Side effects~~

```
var result = Add(5, 2); //7
```

SQUER

Object Oriented vs. Functional Programming

Object Oriented

- Imperative
- Objects + Methods
- Mutable data
- Loops
- Inheritance

Functional

- Declarative
- (pure) Functions
- Immutable data
- Recursion
- Pattern Matching
- Higher-order functions
- Currying

SQUER

Object Oriented vs. Functional Programming

Object Oriented

- Imperative
- Objects + Methods
- Mutable data
- Loops
- Inheritance

Functional

- Declarative
- (pure) Functions
- Immutable data
- Recursion
- Pattern Matching
- Higher-order functions
- Currying

C# !C#

SQUER

Why "Going Functional with C#"?

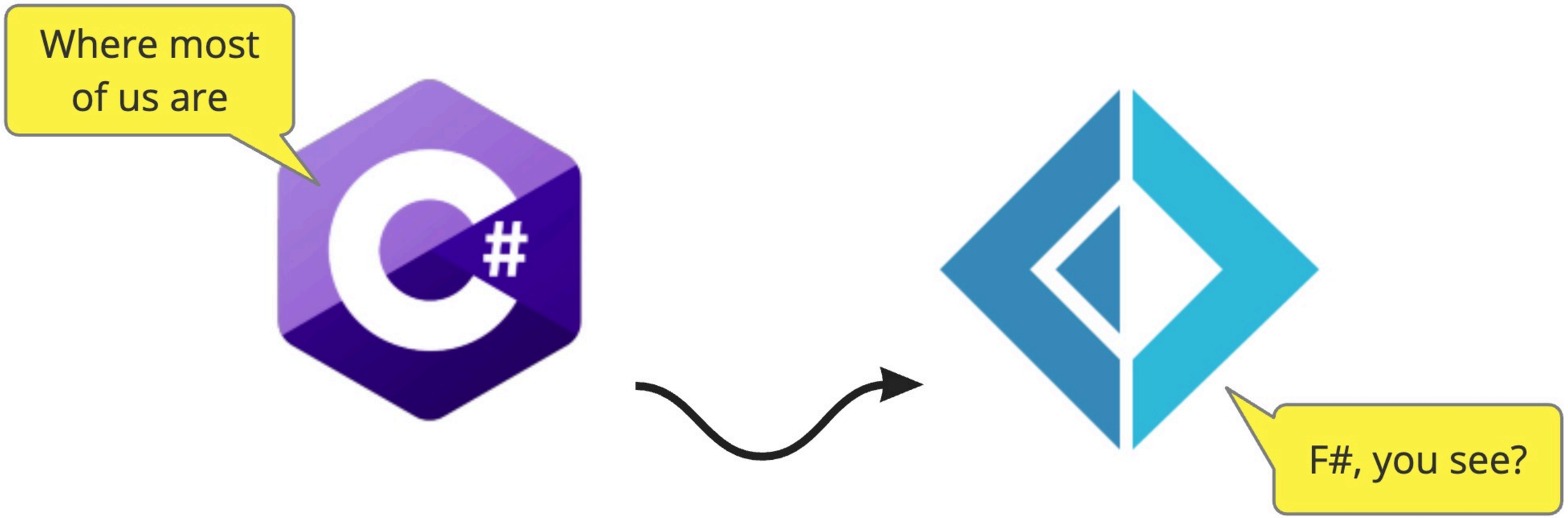
What is C#?

C# is a object-oriented programming language that enables developers to build a variety of secure and robust application that run on the .NET.

https://dotnet.microsoft.com/en-us/learn/csharp

SQUER

Why not Going Full Circle C# -> F#?



SQUER

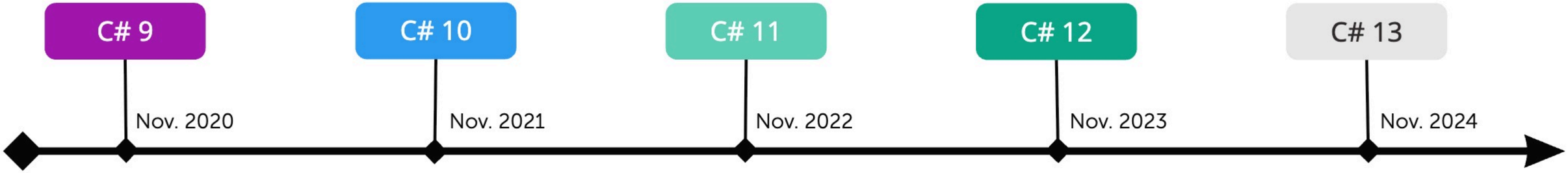
C# Just Object-Oriented?

C# (/ˌsiː ˈʃɑːrp/ *see SHARP*)[b] is a general-purpose high-level programming language supporting multiple paradigms. C# encompasses static typing,[16]:4 strong typing, lexically scoped, imperative, declarative, functional, generic,[16]:22 object-oriented (class-based), and component-oriented programming disciplines.[17]

https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

SQUER

Some Functional Features in C#



- Record Types
- Init Only Setters

- Lambda expression improvements
- Record Structs
- Pattern matching improvements

- List Patterns
- Required Members

- Lambda expression improvements
- Collection Expressions

- 🤔

SQUER

Functions As First-Class Citizens

```
var operation = GetOperation('+');
var result = operation(1, 2); // 3
```

SQUER

Functional? - Functional!

```
public static Func<double, double, double> GetOperation(char @operator)
    => @operator switch
    {
        '+' => (a, b) => a + b,
        '-' => (a, b) => a - b,
        '*' => (a, b) => a * b,
        '/' => (a, b) => a / b,
        _ => throw new ArgumentException("unknown operator")
    };
```

```
var operation = GetOperation('+');
var result = operation(1, 2); // 3
```

SQUER

Another Example

```
var process = GetProcess(config);

var outcome = process(data);
```

SQUER

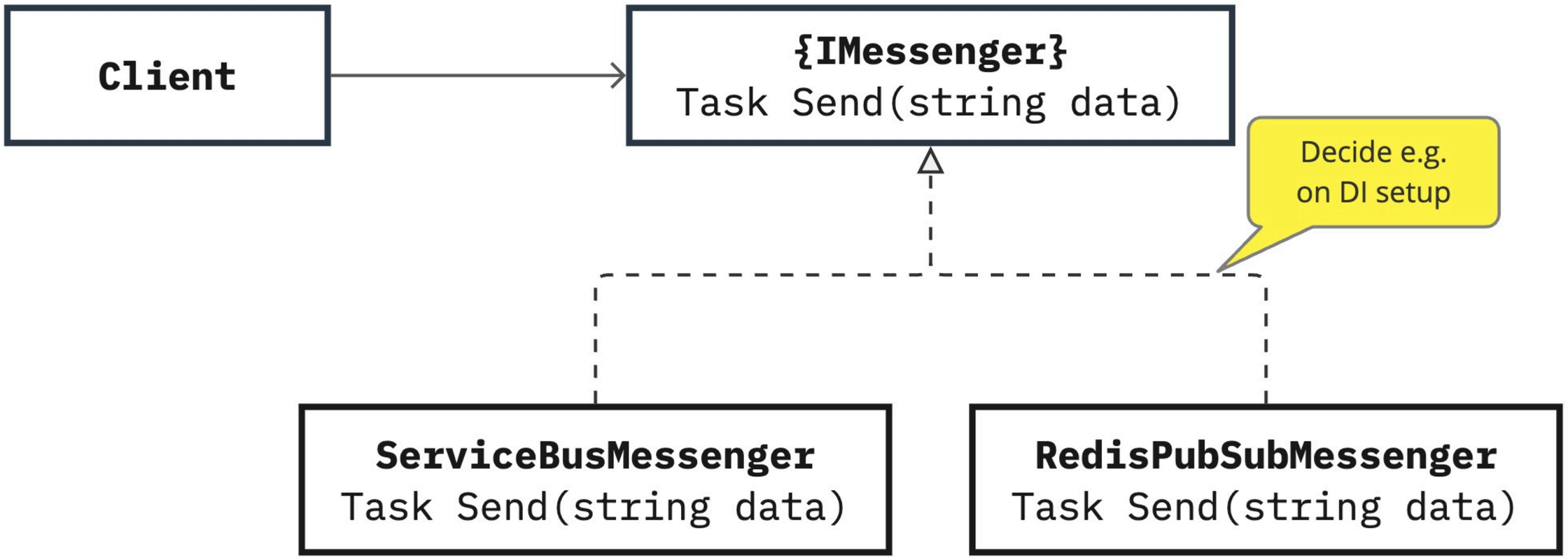
Another Example

```
var process = GetProcess(config);

var outcome = process(data);
```

```
private static Func<byte[], string> GetProcess(ProcessingConfiguration config)
    => config.LegacyProcessing
        ? data =>
        {
            var legacyProcessor = new LegacyProcessor(config, false, "secret");
            legacyProcessor.ClockInit(DateTimeOffset.UtcNow.ToUnixTimeMilliseconds());
            return legacyProcessor.DoMyWeirdThing(data, true, false, 123, true);
        }
        : data => FancyProcessor.PerformAmazingStuff(data, DateTime.UtcNow);
```

SQUER

Polymorphism: Inheritance



SQUER

Pattern Matching

```
public Task Send(string data, MsgConfig config) =>
    config switch
    {
        MsgConfig.RedisPubSub => PublishToRedis(data),
        MsgConfig.ServiceBus => SendViaServiceBus(data),

        _ => throw new ArgumentOutOfRangeException(
            nameof(config), config, null)
    };
```

SQUER

Pattern Matching

```
if (obj is int number)
    return number;
```

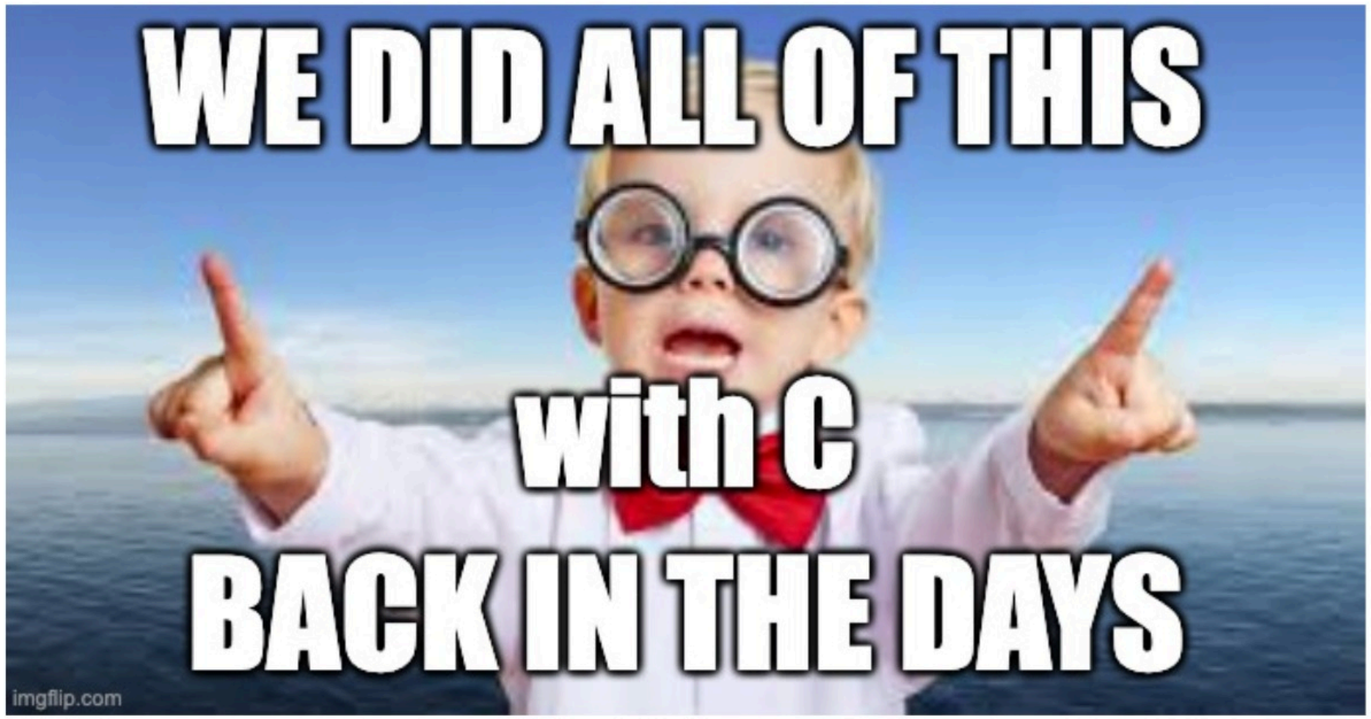
```
if (obj is int negativeNumber and < 0)
    return negativeNumber;
```

```
var isDigit = character is >= '0' and <= '9';
```

```
if (employees is IReadOnlyList<Worker> workers &&
    workers[0] is Electrician electrician)
{
    electrician.ChangeLamp();
}
```

SQUER

But...Wait



SQUER

The Benefit?

https://www.collinsdictionary.com/dictionary/english/expressive

SQUER

Pattern Matching

```
var personInfo = person switch
{
    Worker { Profession: var profession }
        => $"works as {profession}",
    Employee { IsActive: true, Department: var department }
        => $"works in {department}",
    Employee { IsActive: false, Boss: var lastBoss }
        => $"was working for {lastBoss}",
    { Hobby: var hobby }
        => $"likes {hobby}",
};
```

SQUER

🚂 Chew choo! 🚂
Let me take you on a Journey

SQUER

Customer

```
public IActionResult GetCustomer(string name)
{
    Customer customer = _customerService.Find(name);
    return Ok(customer);
}
```

SQUER

What if there is no Customer?

SQUER

Customer?

```
public IActionResult GetCustomer(string name)
{
    Customer? customer = _customerService.Find(name);
    if (customer == null)
    {
        _logger.LogInformation($"Customer {name} was not found");
        return NoContent();
    }
    return Ok(customer);
}
```

SQUER

Maybe<Customer>

```
public IActionResult GetCustomer(string name)
{
    Maybe<Customer> customer = _customerService.Find(name);
    if (customer.HasNoValue)
    {
        _logger.LogInformation($"Customer {name} was not found");
        return NoContent();
    }
    return Ok(customer.Value);
}
```

SQUER

What if the database is down?

SQUER

Error case?

```
try
{
    Maybe<Customer> customer = _customerService.Find(name);
    if (customer.HasNoValue)
    {
        _logger.LogInformation($"Customer {name} was not found");
        return NoContent();
    }
    return Ok(customer.Value);
}
catch (Exception e)
{
    _logger.LogError(e.Message);
    return Problem(e.Message);
}
```

SQUER

Let's try again...

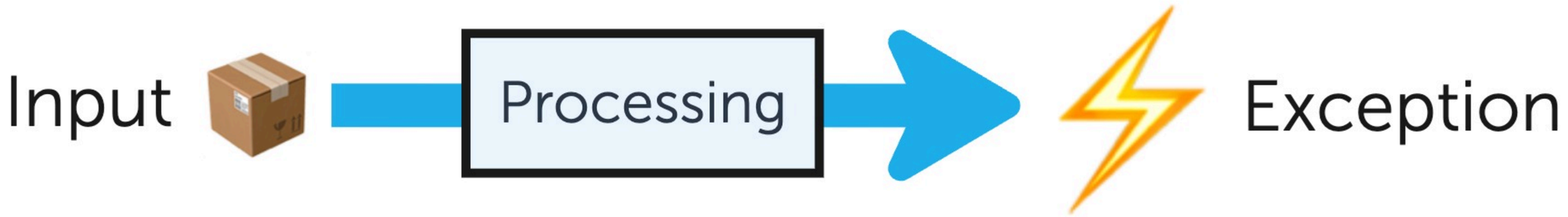
Normal Operation

```
var sum = Divide(1, 2);
```

SQUER

Problematic Operation (Classic)

```
var sum = Divide(1, 0);
```



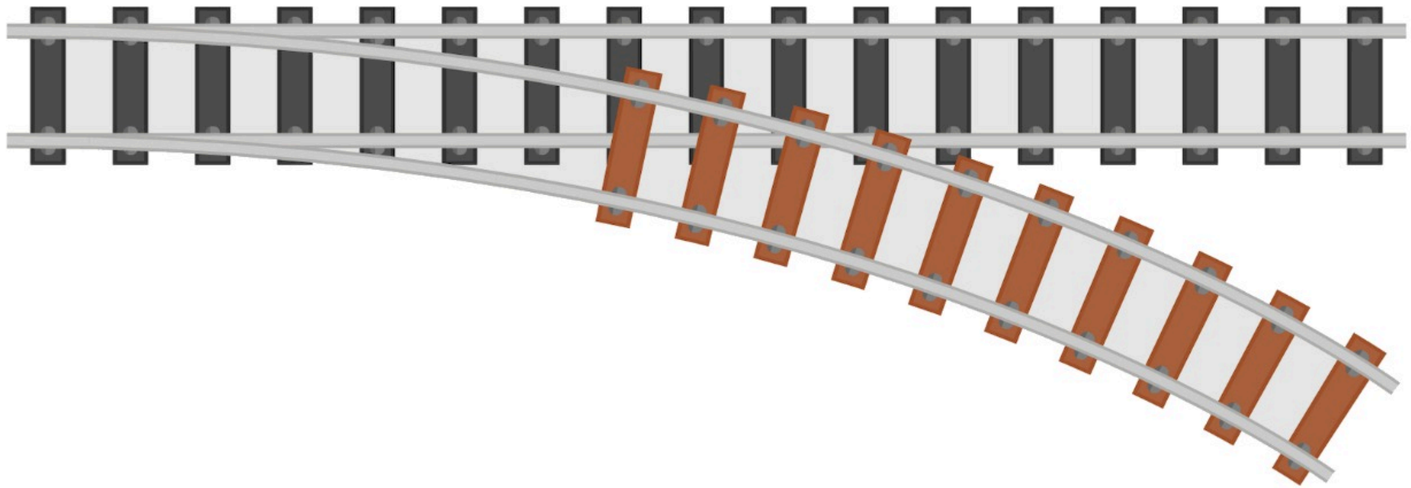
SQUER

SQUER

https://www.tapetenwelt.com/fototapete-lokomotive-thomas

SQUER

Railway Oriented Programming



Success Track

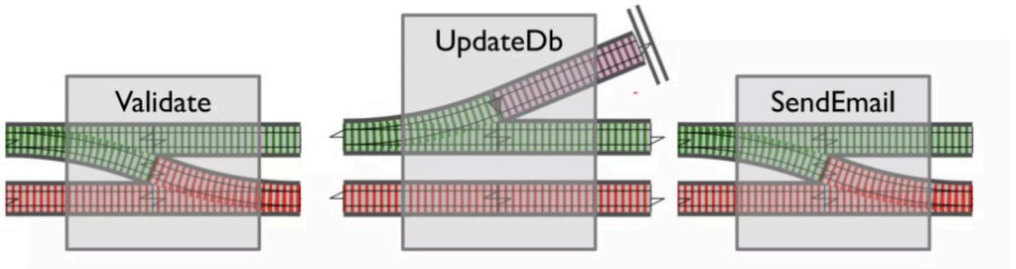
Failure Track

SQUER

Image designed by Freepik

Railway Oriented Programming

Scott Wlaschin

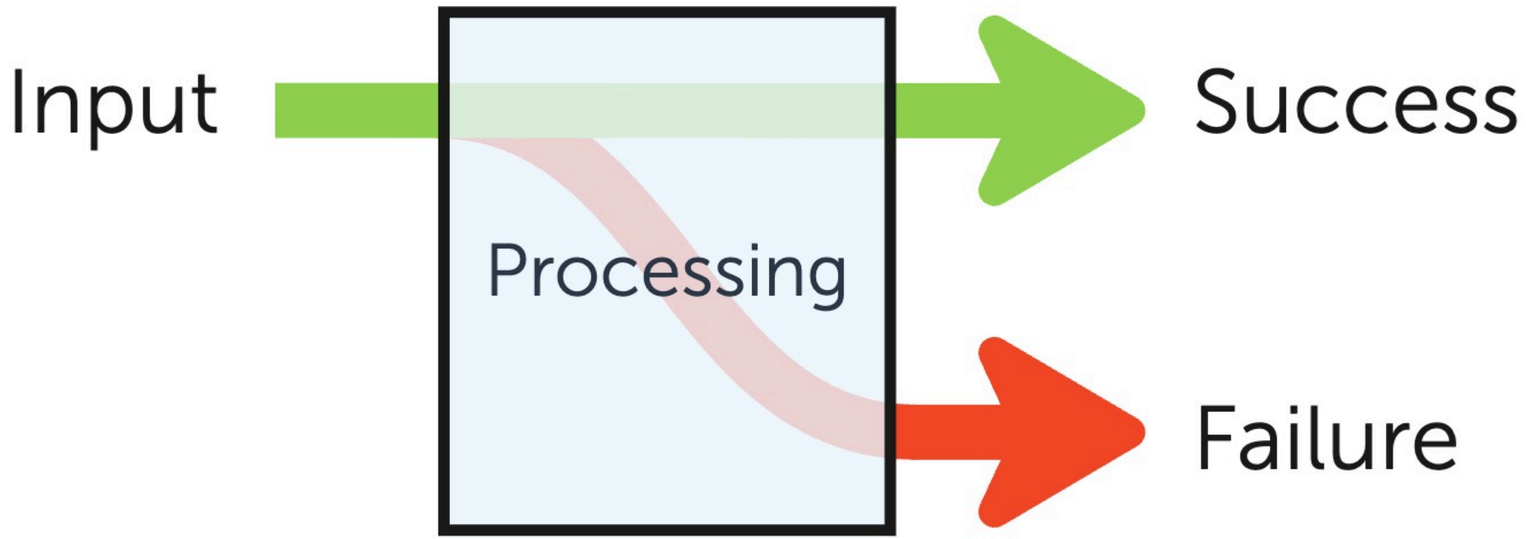


https://fsharpforfunandprofit.com

F# for Fun and Profit

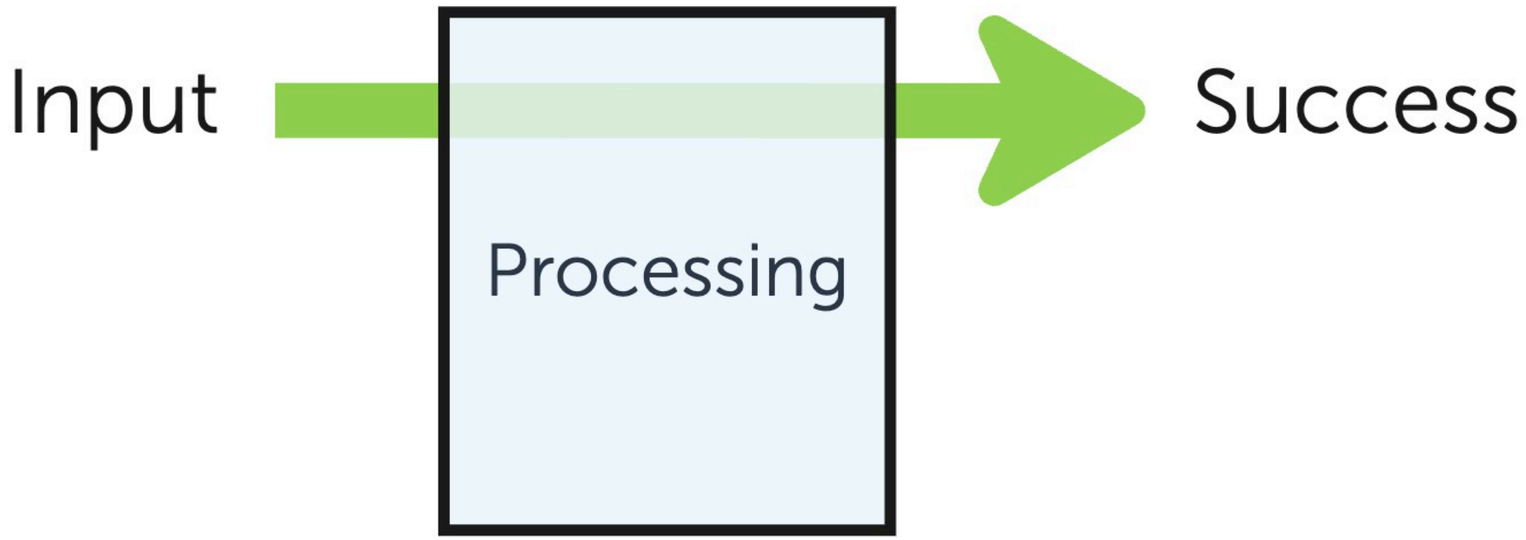
SQUER

Railway Oriented



SQUER

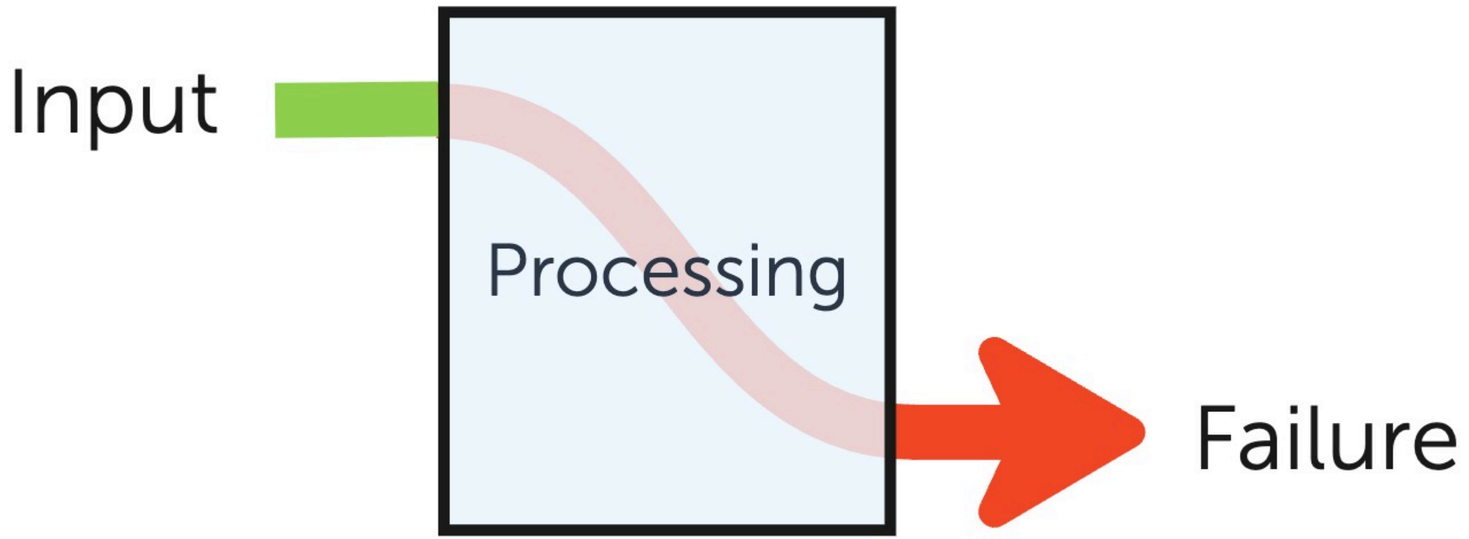
Normal Operation (Railway Oriented)




```
var sum = Divide(1, 2);
```

SQUER

Problematic Operation (Railway Oriented)




```
var sum = Divide(1, 0);
```

SQUER

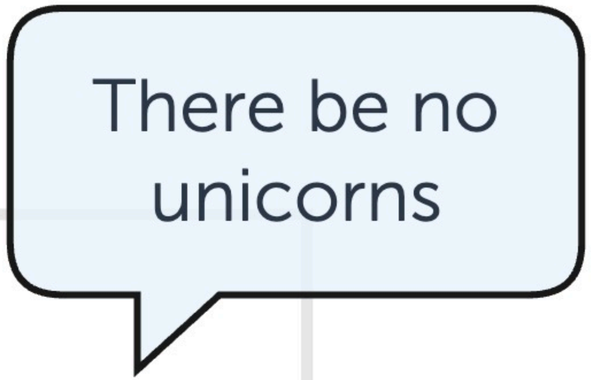
Changed Signature - Result<T>

```
private static Result<int> Divide(int n, int d)
```

SQUER

Result<T>

API of some fictitious library




```
class Result<T>
{
    public T Value { get; }
    public string Error { get; }
    public bool IsSuccess => Error == null;
    public bool IsFailure => !IsSuccess;
}
```

SQUER

Result<T, TError>

API of some fictitious library

```
class Result<T, TError>
{
    public T Value { get; }
    public TError Error { get; }
    public bool IsSuccess => Error == null;
    public bool IsFailure => !IsSuccess;
}
```

SQUER

Working with Result - Classic

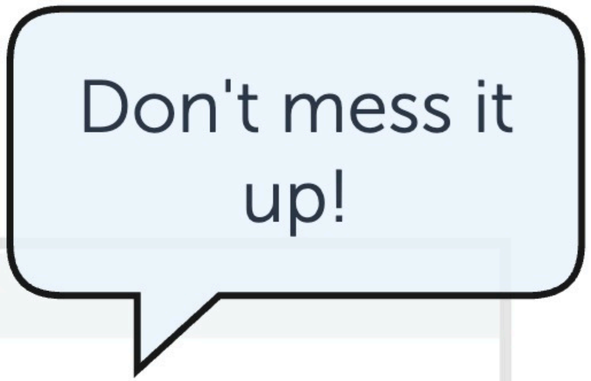
API of some fictitious library

```
var result = Divide(Some, All);

if (result.IsSuccess)
    Console.WriteLine($"The ratio is {result.Value}.");
else
    Console.WriteLine($"Could not divide: {result.Error}");
```

SQUER

Working with Result - Classic

API of some fictitious library




```
var result = Divide(Some, All);

if (result.IsSuccess)
    Console.WriteLine($"The ratio is {result.Value}.");
else
    Console.WriteLine($"Could not divide: {result.Error}");
```

SQUER

Working with Result - Callbacks

API of some fictitious library

```
var result = Divide(some, all);

result.OnSuccess(ratio => Console.WriteLine($"The ratio is {ratio}."));
result.OnFailure(error => Console.WriteLine($"Could not divide: {error}"));
```

SQUER

Working with Result - Callbacks Fluent

API of some fictitious library

```
Divide(some, all)
    .OnSuccess(ratio => Console.WriteLine($"The ratio is {ratio}."))
    .OnFailure(error => Console.WriteLine($"Could not divide: {error}"));
```

SQUER

Working with Result - Callbacks Fluent 2

API of some fictitious library

```
Divide(some, all)
    .Match(
        ratio => Console.WriteLine($"The ratio is {ratio}."),
        error => Console.WriteLine($"Could not divide: {error}")
    );
```

SQUER

SQUER

Vladimir Khorikov
vkhorikov

Downloads Full stats →

Total **22.5M**

Current version **100.7K**

Per day average **6.9K**

About

Last updated 2 months ago

Functional Extensions for C# aka **CSharpFunctionalExtensions**

https://github.com/vkhorikov/CSharpFunctionalExtensions

SQUER

Result<Customer> in Controller - Classical

API: CSharpFunctionalExtensions

```
public IActionResult GetCustomer(string name)
{
    Result<Customer> result = _customerService.Find(name);
    if (result.IsFailure)
    {
        _logger.LogInformation(result.Error);
        return Problem(result.Error);
    }

    return Ok(result.Value);
}
```

SQUER

Result<Customer> in Controller - Fluent

API: CSharpFunctionalExtensions

```
public IActionResult GetCustomer(string name)
{
    return _customerService.Find(name)
        .TapError(error => _logger.LogError(error))
        .Match(
            customer => Ok(customer),
            error => Problem(error)
        );
}
```

SQUER

Result<Customer> in Controller - Not Found & Error

API: CSharpFunctionalExtensions

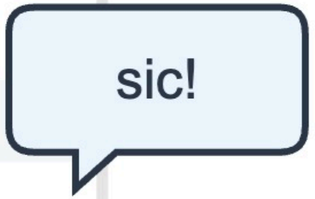
```
public IActionResult GetCustomer(string name)
{
    Result<Maybe<Customer>> result = _customerService.Find(name);
    if (result.IsFailure)
    {
        _logger.LogError(result.Error);
        return Problem(result.Error);
    }

    var maybeCustomer = result.Value;
    if (maybeCustomer.HasNoValue)
    {
        _logger.LogInformation($"Customer {name} was not found");
        return NoContent();
    }

    return Ok(maybeCustomer.Value);
}
```

SQUER

Result<Customer> in Controller - Not Found & Error

API: CSharpFunctionalExtensions

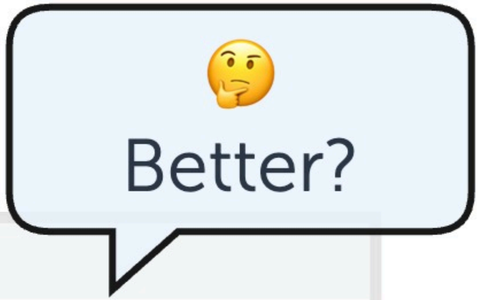
```
public IActionResult GetCustomer(string name)
{
    return _customerService.Find(name)
        .TapError(error => _logger.LogError(error))
        .Match(
            maybeCustomer => maybeCustomer
                .Match<IActionResult, Customer>(
                    customer => Ok(customer),
                    () => NoContent()
                ),
            error => Problem(error)
        );
}
```



SQUER

Result<Customer> in Controller - Shorthands

API: CSharpFunctionalExtensions

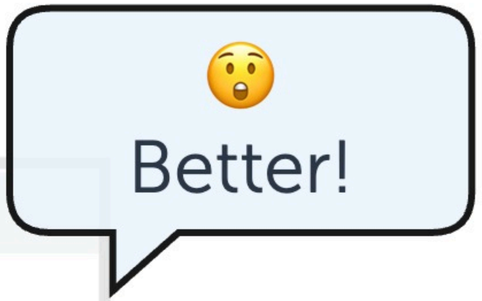



```
public IActionResult GetCustomer(string name)
    => _customerService.Find(name)
        .TapError(error => _logger.LogError(error))
        .Match(maybeCustomer => maybeCustomer
                .Match<IActionResult, Customer>(Ok, NoContent),
            error => Problem(error));
```

SQUER

Result<Customer> in Controller - Just Shorthands

API: CSharpFunctionalExtensions




```
public IActionResult GetCustomer(string name)
    => _customerService.Find(name)
        .TapError(LogError)
        .Match(maybeCustomer => maybeCustomer
                .Match<IActionResult, Customer>(
                    Ok,
                    NoContent),
            Problem);
```

SQUER

.Map(value => ...otherValue)

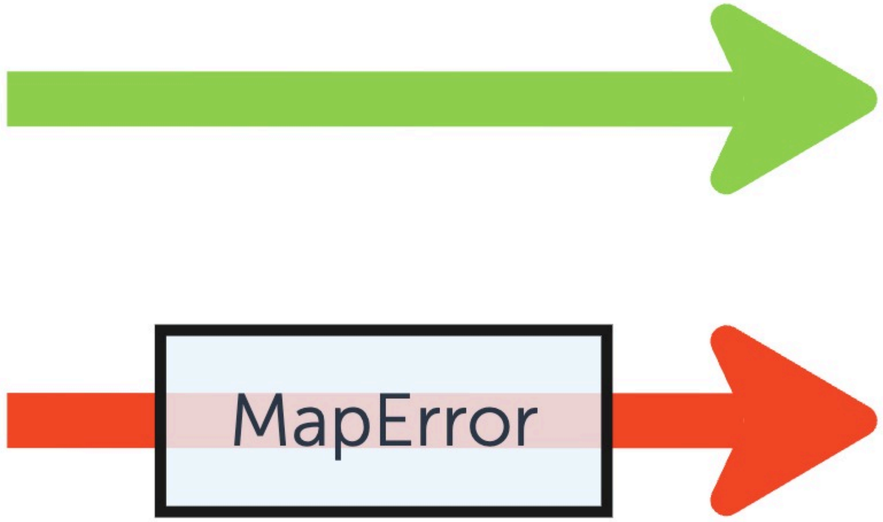
API: CSharpFunctionalExtensions


```
.Map(customer => customer.MapToDto())
```

SQUER

.MapError(error => ...otherError)

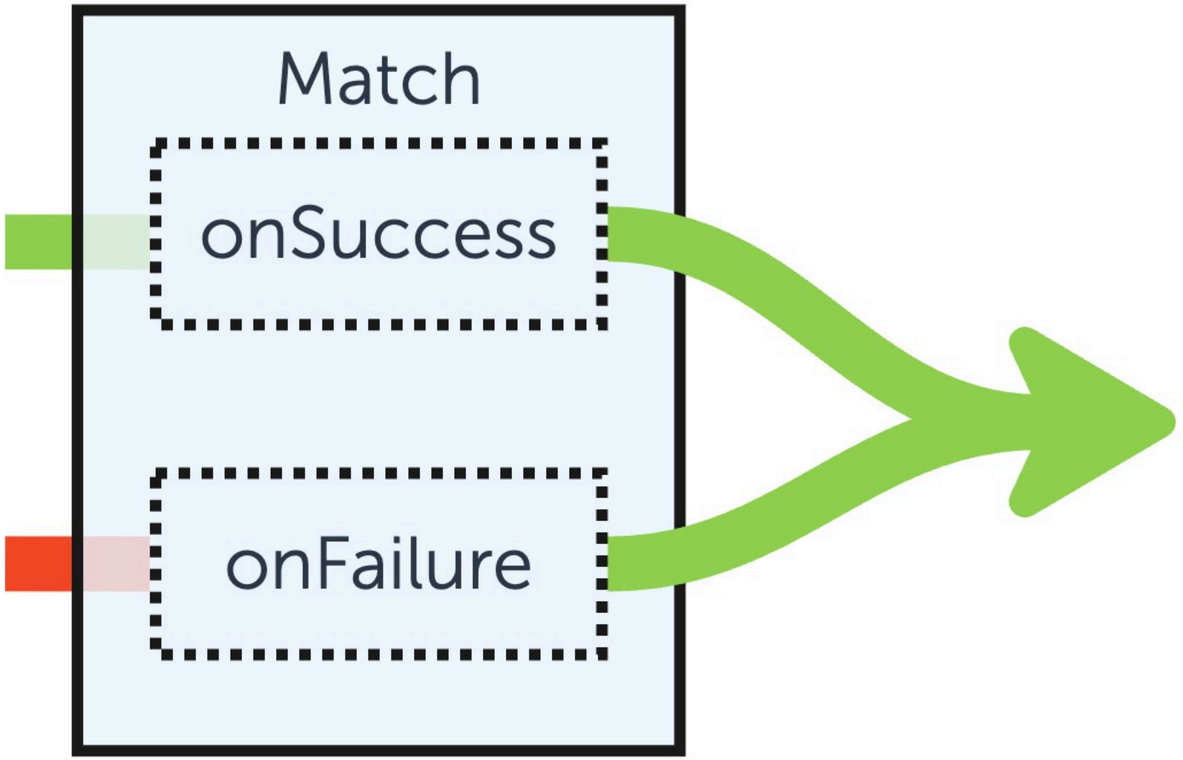
API: CSharpFunctionalExtensions



SQUER

.Match(value => ..., error => ...)

API: CSharpFunctionalExtensions

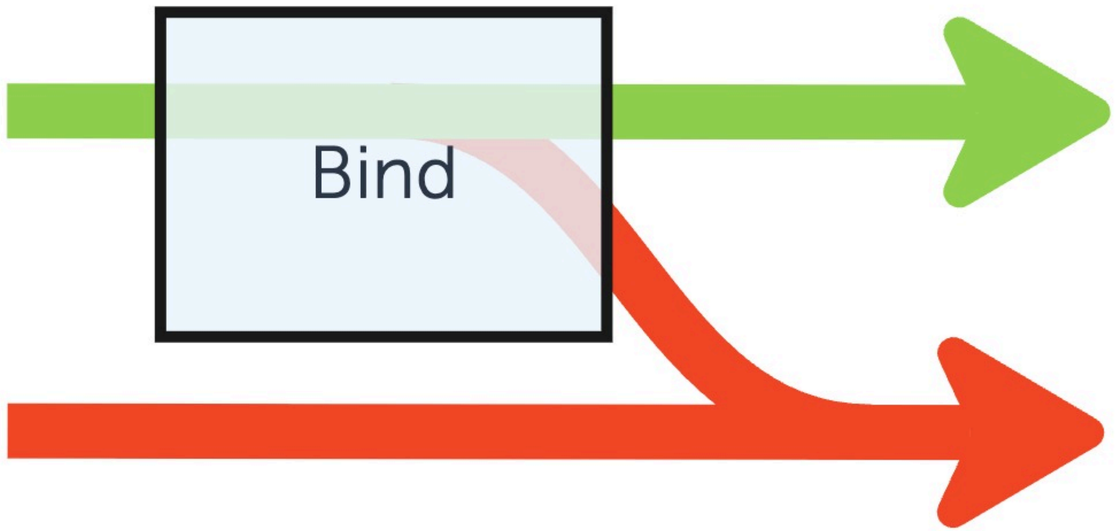



```
.Match(customer => Ok(customer),
    error => Problem(error));
```

SQUER

.Bind(value => ...otherResult)

API: CSharpFunctionalExtensions

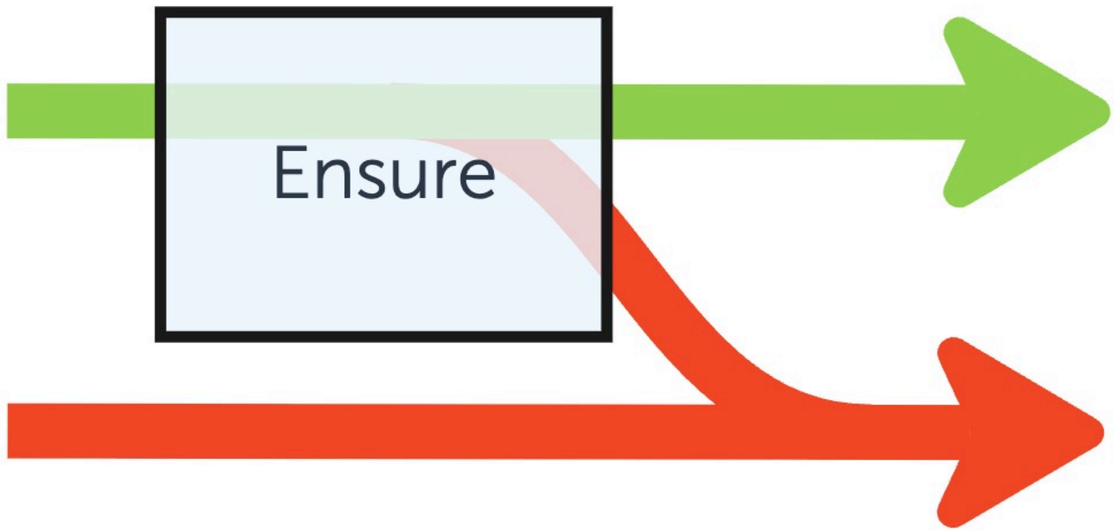



```
.Bind(schedule => _scheduleSender.SendSchedule(schedule))
```

SQUER

.Ensure(value => ...bool, error)

API: CSharpFunctionalExtensions

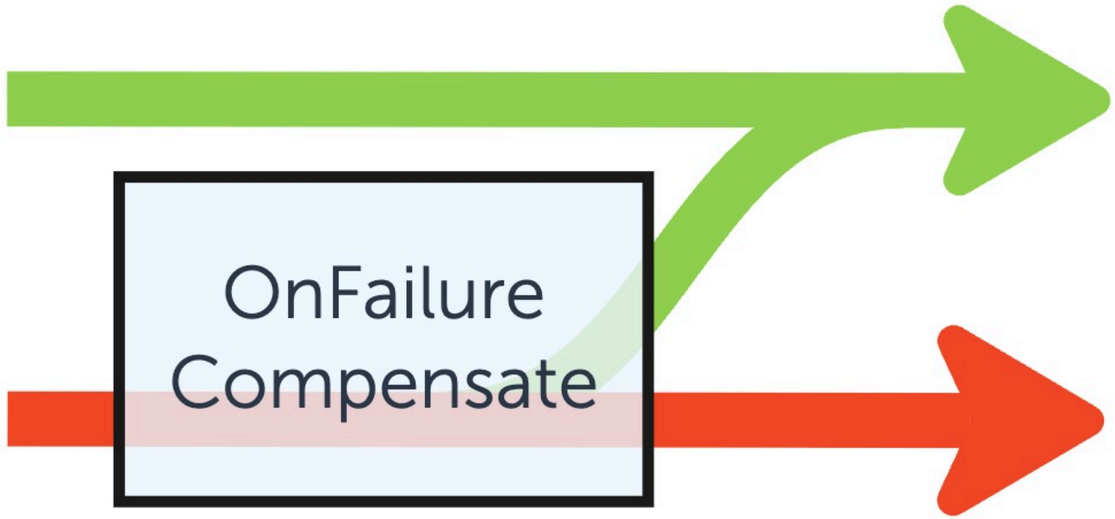



```
.Ensure(IsValid, "Invalid Schedule!")
```

SQUER

.OnFailureCompensate(error => ...otherResult)

API: CSharpFunctionalExtensions

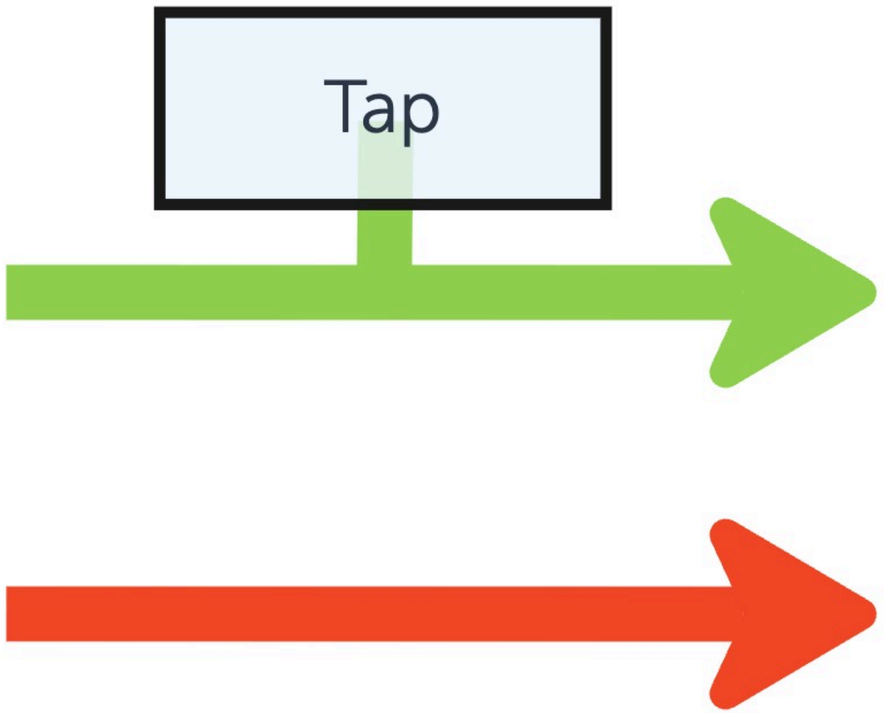



```
.OnFailureCompensate(_ => Maybe<DefaultControlInformation>.None)
```

SQUER

.Tap(value => { ... })

API: CSharpFunctionalExtensions

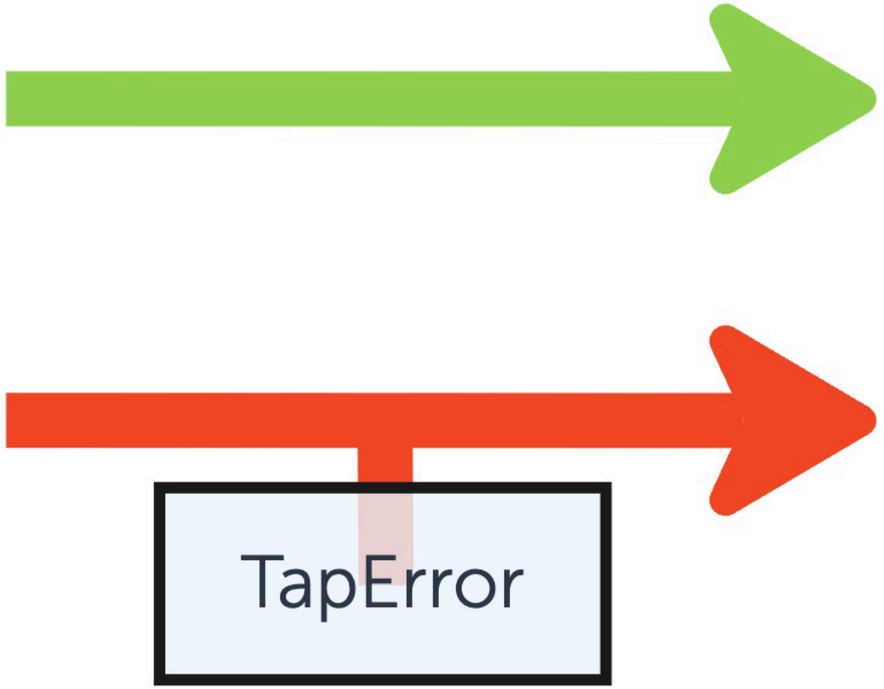



```
.Tap(optimizedSchedule =>
  _logger.LogInformation("Optimized schedule: {optimizedSchedule}", optimizedSchedule))
```

SQUER

.TapError(error => { ... })

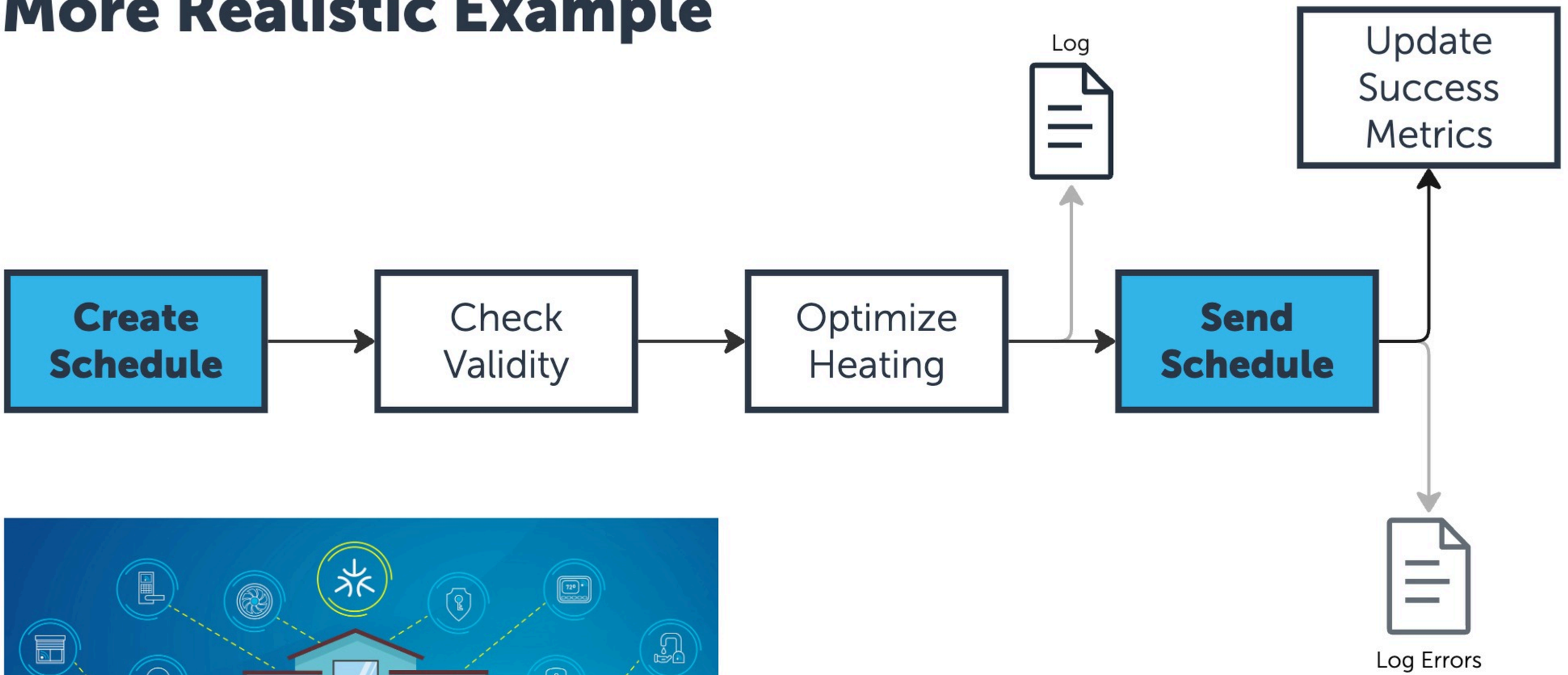
API: CSharpFunctionalExtensions




```
.TapError(error => _logger.LogError(error))
```

SQUER

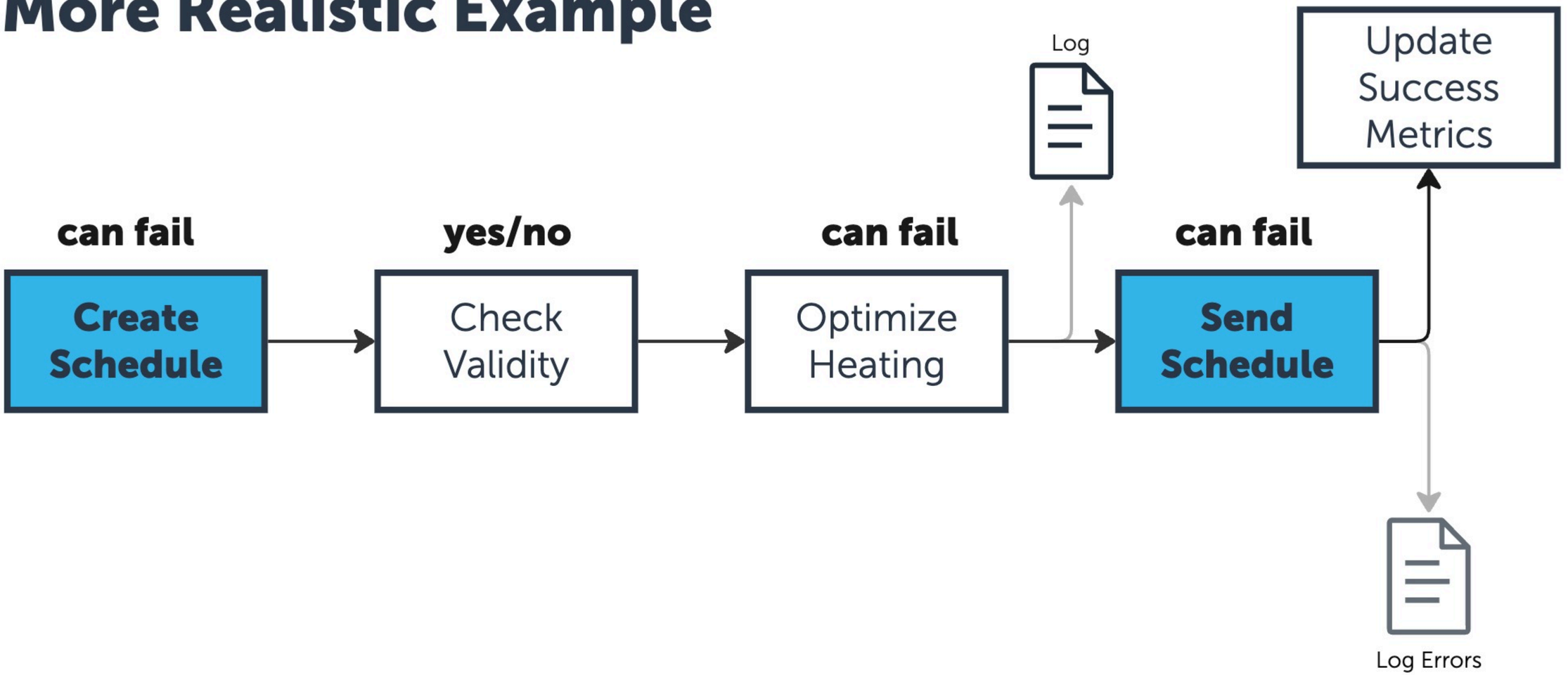
A More Realistic Example



https://www.flickr.com/photos/smarthomeperfected/51255561279

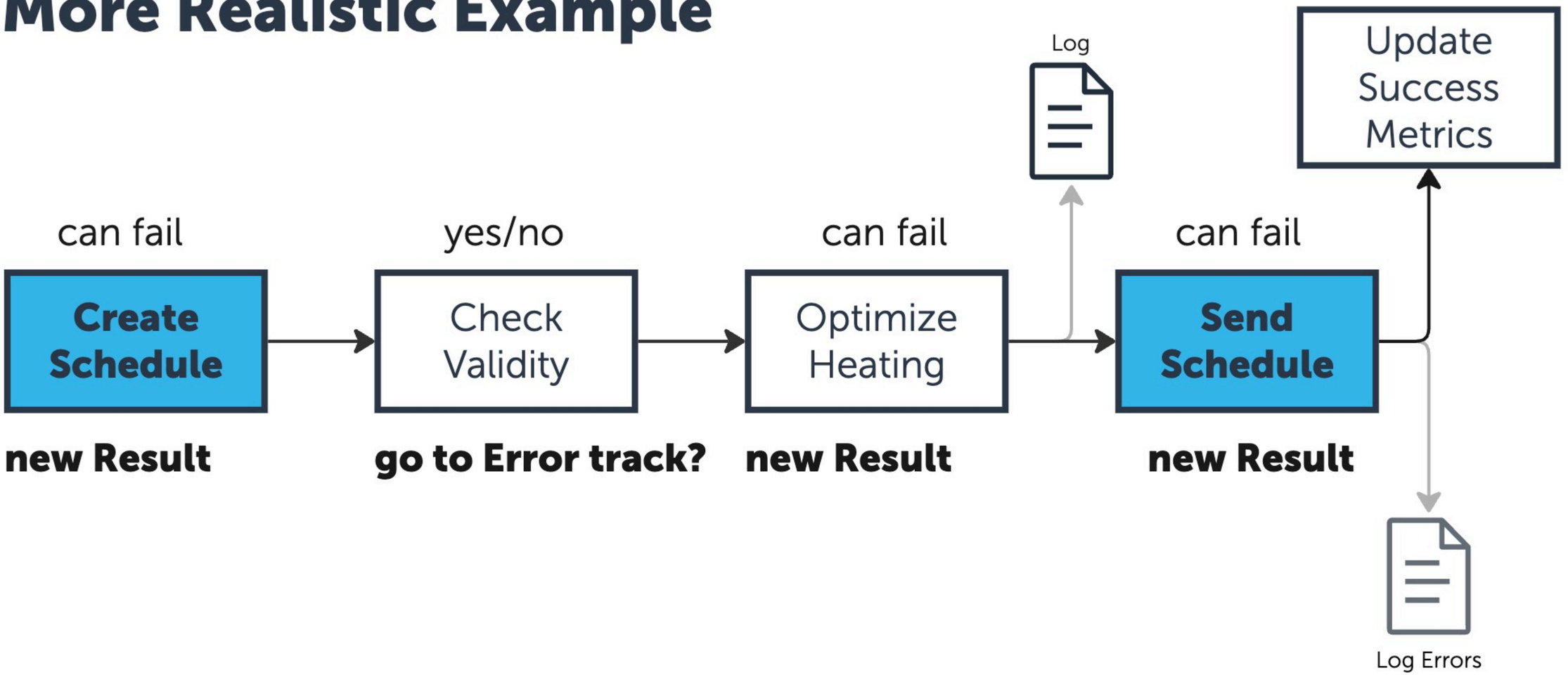
SQUER

A More Realistic Example



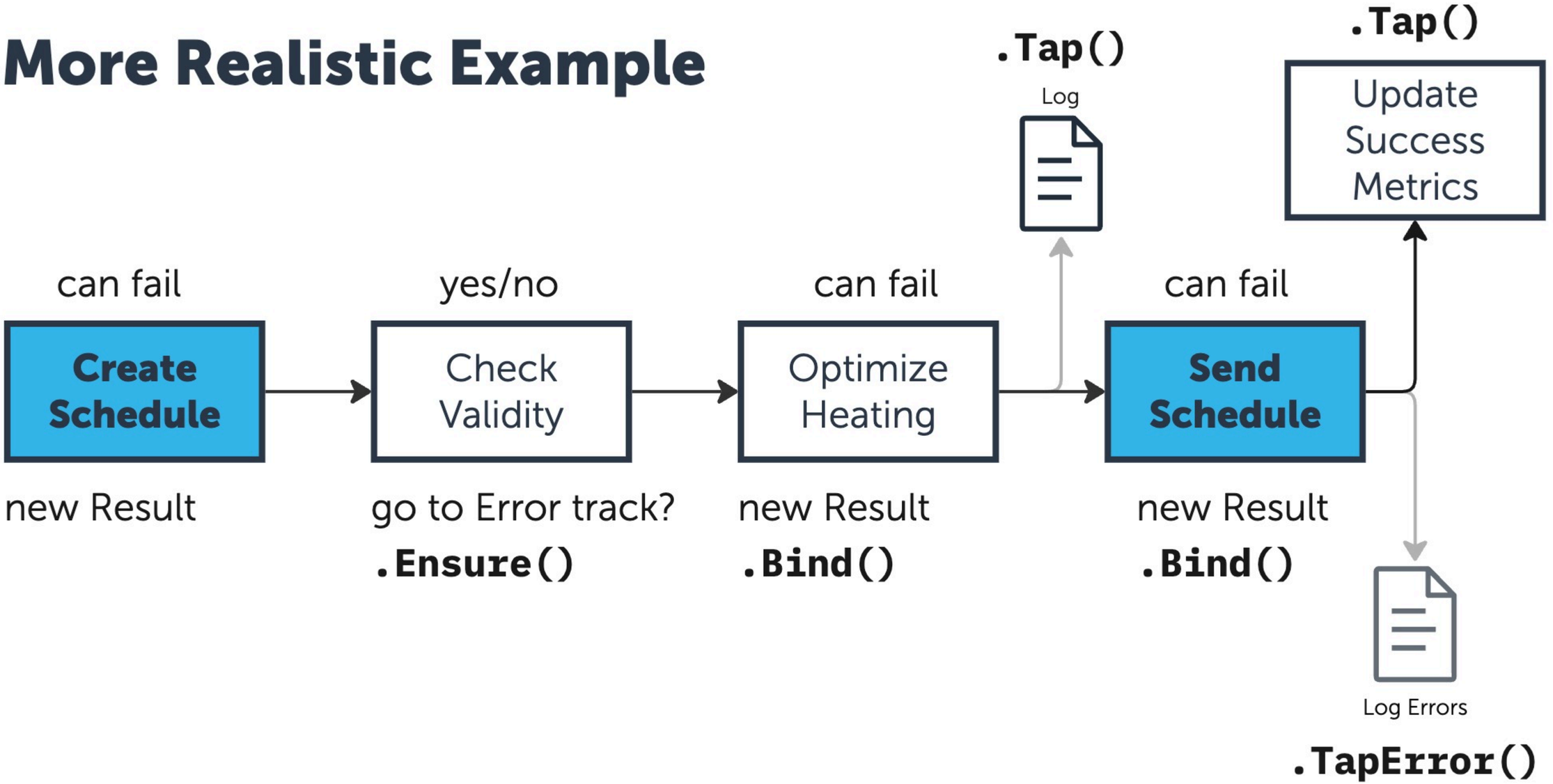
SQUER

A More Realistic Example



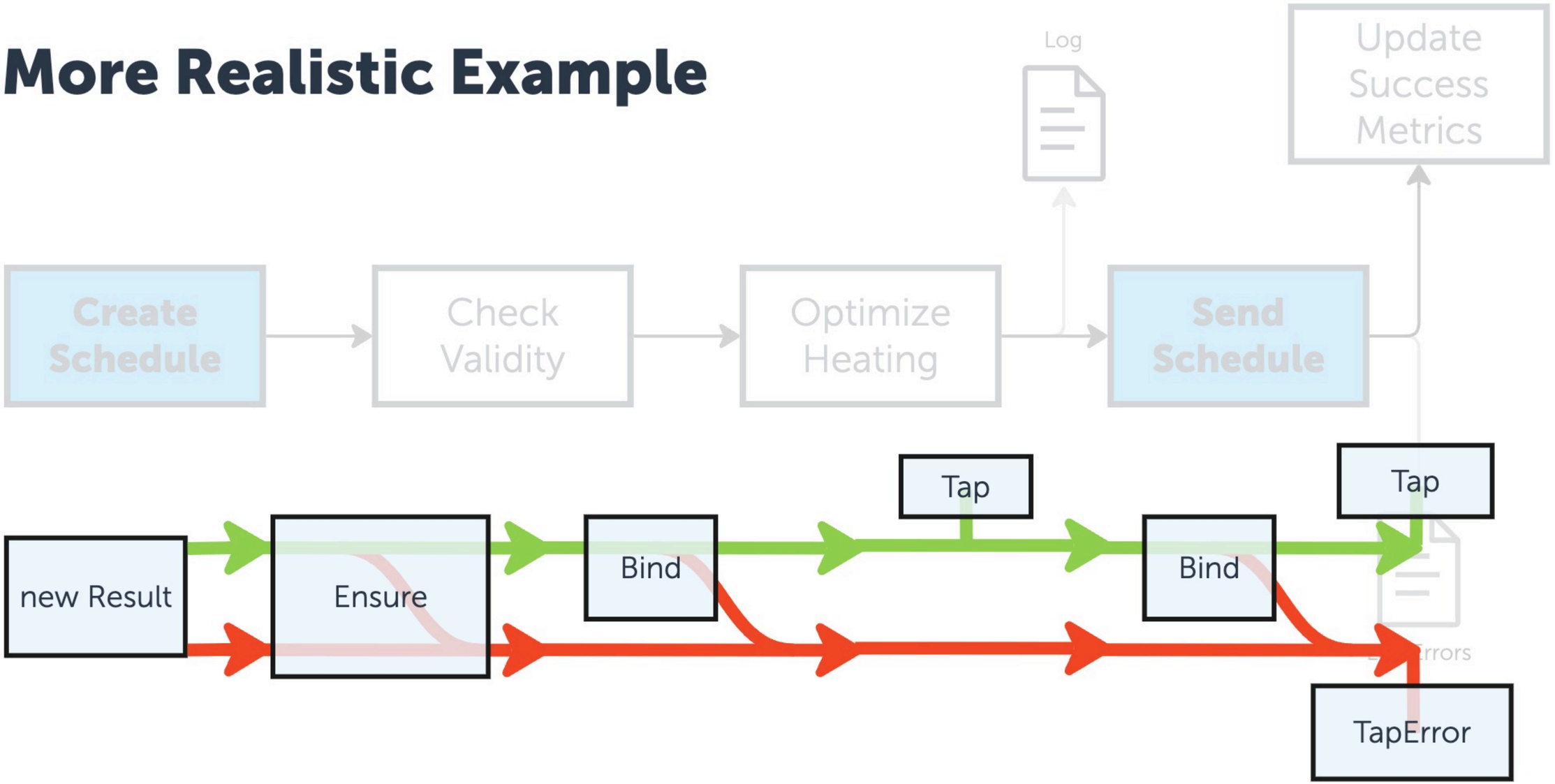
SQUER

A More Realistic Example



SQUER

A More Realistic Example



SQUER

A More Realistic Example - The Code

```
return await _scheduleCreator.Create(Id, controls)
  .Ensure(IsValid, "Invalid Schedule!")
  .Bind(OptimiseHeatingControls)
  .Tap(optimizedSchedule =>
    _logger.LogInformation("Optimized schedule: {optimizedSchedule}", optimizedSchedule))
  .Bind(schedule => _scheduleSender.SendSchedule(schedule))
  .TapIf(sendScheduleResult => sendScheduleResult.IsSent,
    () => _successfullSend++)
  .OnFailure(error =>
    _logger.LogError("Sending failed! Details: {ErrorMessage}", error));
```

SQUER


Insights From
the Real World
https://www.pexels.com/de-de/foto/low-angle-foto-von-dinosaurier-essen-baby-dinosaurier-410857/
SQUER

A Success Story?

SQUER

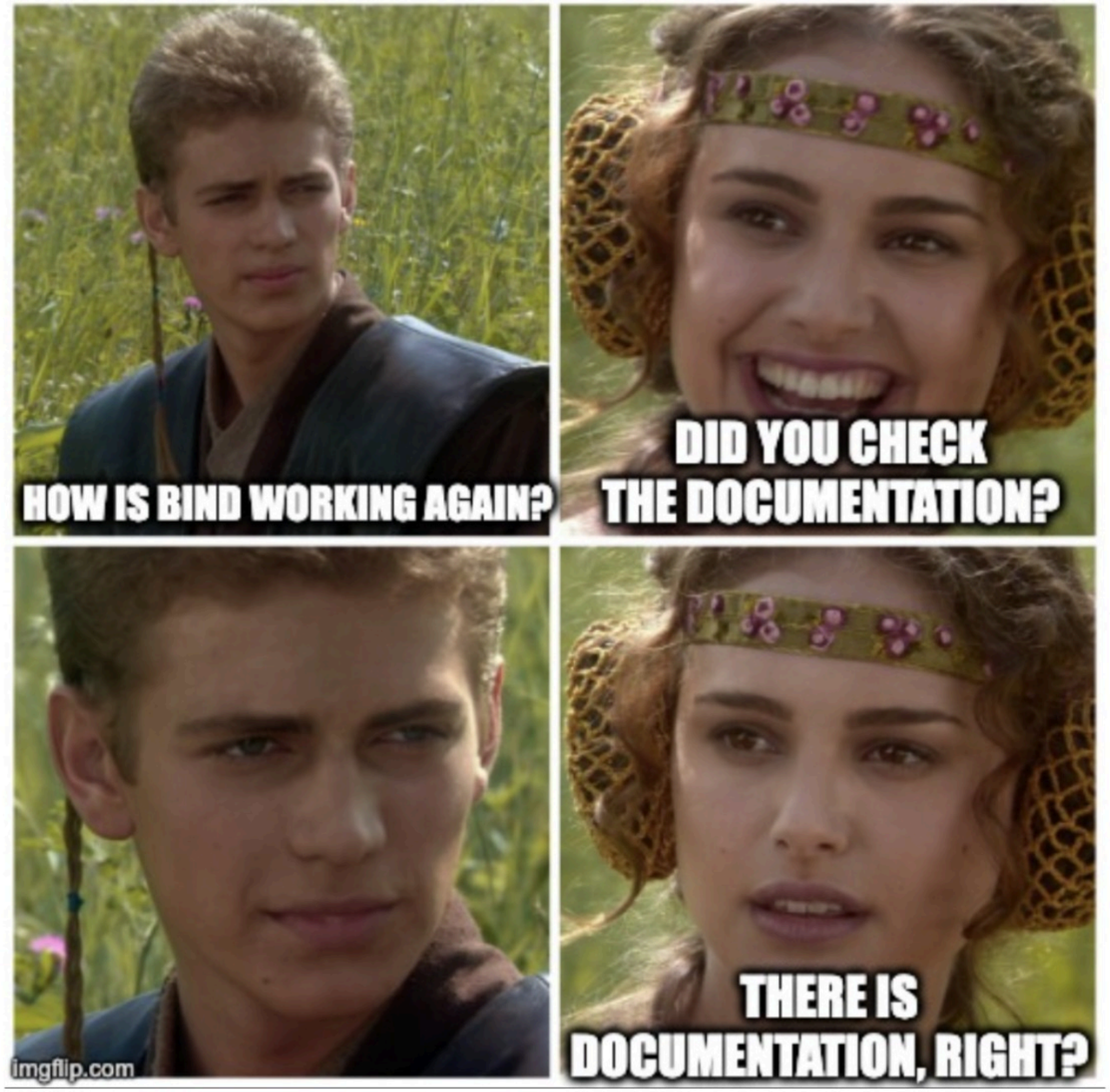
Authoring can be a Pain

```
public IActionResult GetCustomer(string name)
{
    return _customerService.Find(name)
        .TapError(error => _logger.LogError(error))
        .Match(
            maybeCustomer => maybeCustomer.Match(
                customer => Ok(customer),
                () => NoContent()),
            error => Problem(error)
            )
        // .Map(customer => customer)
        // .Ensure(customer => customer.IsVip, "customer is not a VIP")
        // .Match(
        //     customer => Ok(customer),
        //     error => Problem(error)
        // )
        ;
}
```

Many overloads

literal / Result / async / additional parameters

SQUER


HOW IS BIND WORKING AGAIN?
DID YOU CHECK THE DOCUMENTATION?
imgflip.com
THERE IS DOCUMENTATION, RIGHT?

SQUER

Structured Logging from the Error Track

Details are lost when logging should be moved to the caller.

```
if (cannotProcess)
{
    _logger.LogError("Cannot process {InvalidTime} because of {Reason}",
        timeToProcess, reason);
    return Result.Failure<int>(reason);
}
```

- Consider custom error type?
- Keep logging close to occurrence.

SQUER

Co-Existence of Railway Oriented and Classical

? Anti-Pattern
Code Smell

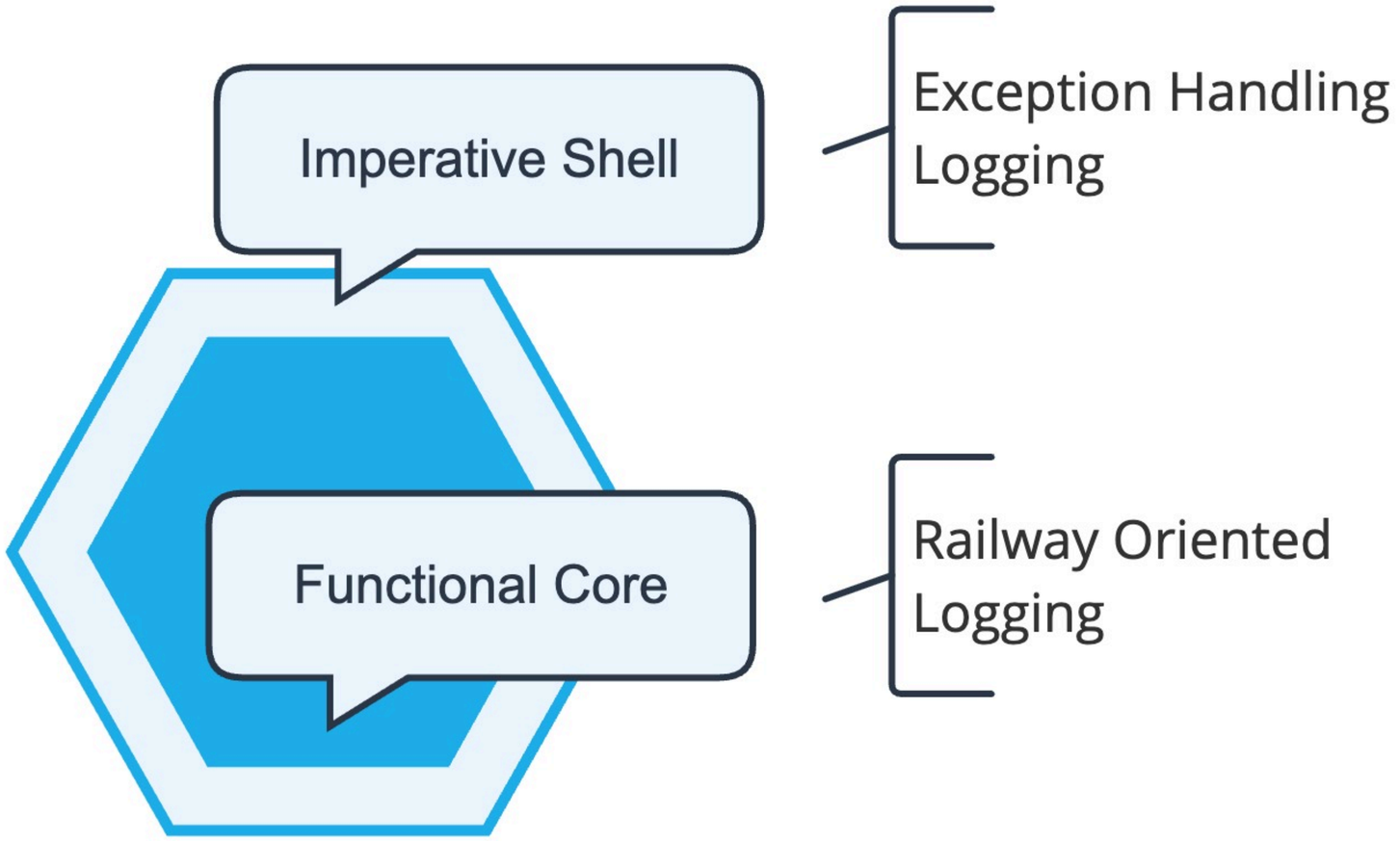
```
try
{
    return _customerService.Find(name)
        .Match(
            customer => Ok(customer),
            error => Problem(error)
        );
}
catch (Exception xcp)
{
    return Problem(xcp.Message);
}
```

Returning a Result<T> does not prevent throwing exceptions

SQUER

Imperative Shell - Functional Core

Ideally



SQUER

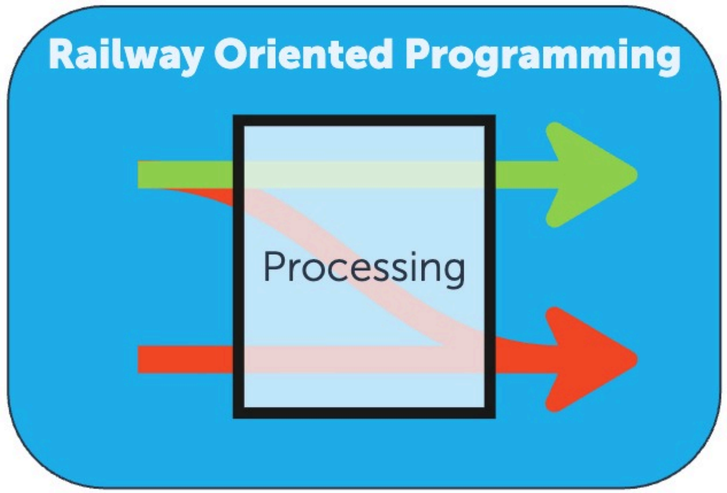
https://www.rawpixel.com/image/3285658/free-photo-image-cafe-bar-bottle

What have we Learned?

Design with **failure case in mind**

Best used with the **whole chain** of flow

Helps to get **terse code**



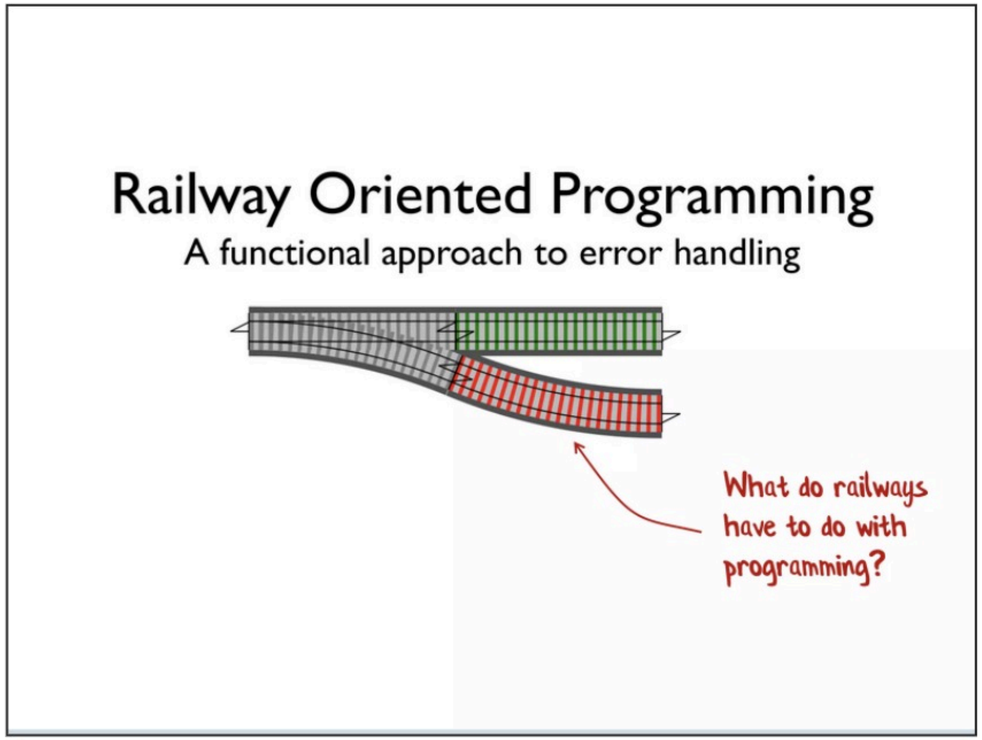
Encourages you to **think about** a **exception** handling strategy

Code might better **tell the story** of a complicated flow

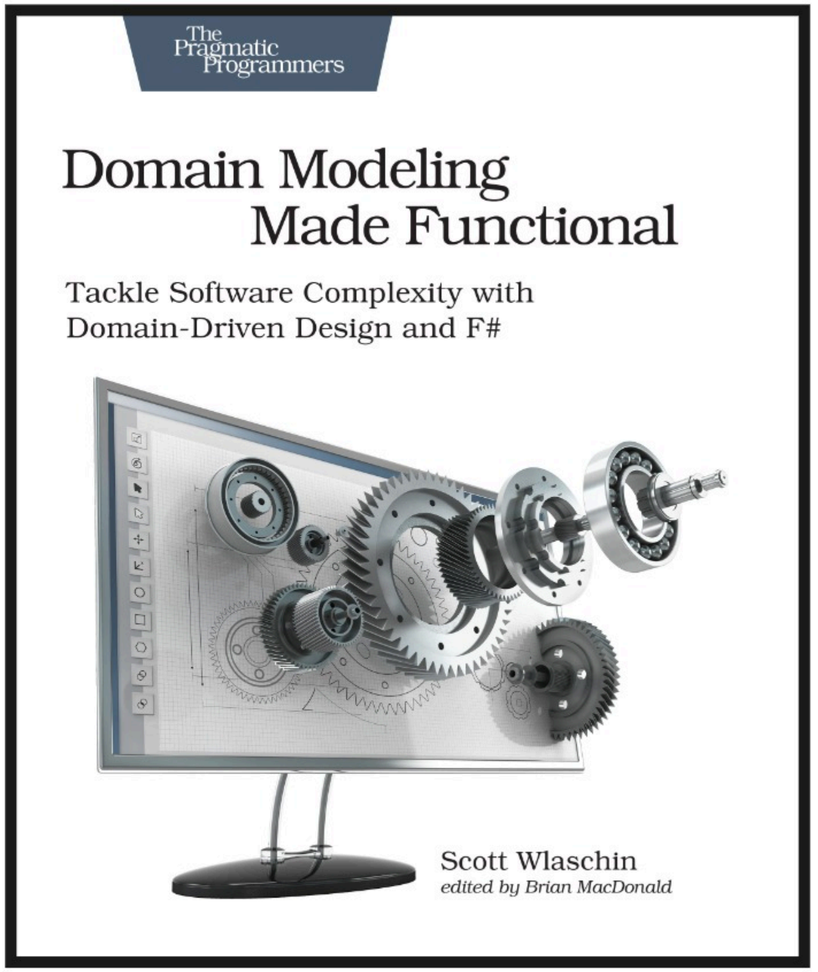
Might be **challenging** to author

SQUER

Resources



https://fsharpforfunandprofit.com/rop/



SQUER

STAY UP TO DATE

Follow us on Youtube, LinkedIn and Instagram

YouTube

LinkedIn

Instagram

SQUER

Paul Rohorzka

@paulroho

paul.rohorzka@squer.io

Your Questions?

SQUER