RAILWAY ORIENTED PROGRAMMING

EIN FUNKTIONALER ANSATZ FÜRS ERROR HANDLING

Patrick Drechsler

red·head·s
SOFTWARE SOLUTIONS SUPPORT

♫ ...HAPPY PATH... ♫

Product Owner: "5 Zeilen Code..."

```
[HttpPost]
public HttpResponseMessage CreateCustomer(string name, string billingInfo)
{
    Customer customer = new Customer(name);

    _repository.Save(customer);

    _paymentGateway.ChargeCommission(billingInfo);

    _emailSender.SendGreetings(name);

    return new HttpResponseMessage(HttpStatusCode.OK);
}
```

red·head·s
SOFTWARE SOLUTIONS SUPPORT

HERKÖMMLICHE FEHLERBEHANDLUNG

```
[HttpPost]
public HttpResponseMessage CreateCustomer(string name, string billingInfo)
{
    // try/catch...
    Customer customer = new Customer(name);
    // null check..

    // try/catch...
    _repository.Save(customer);

    // try/catch...
    _paymentGateway.ChargeCommission(billingInfo);

    // try/catch...
    _emailSender.SendGreetings(name);

    return new HttpResponseMessage(HttpStatusCode.OK);
}
```

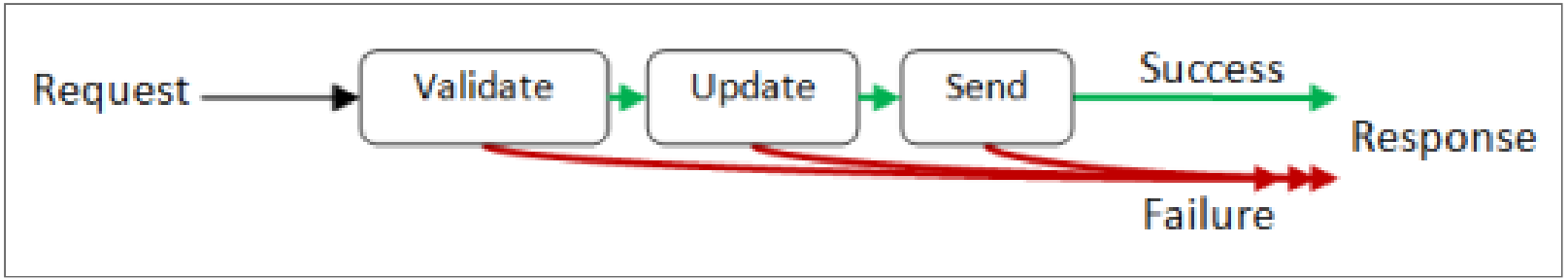
red·head·s
SOFTWARE SOLUTIONS SUPPORT

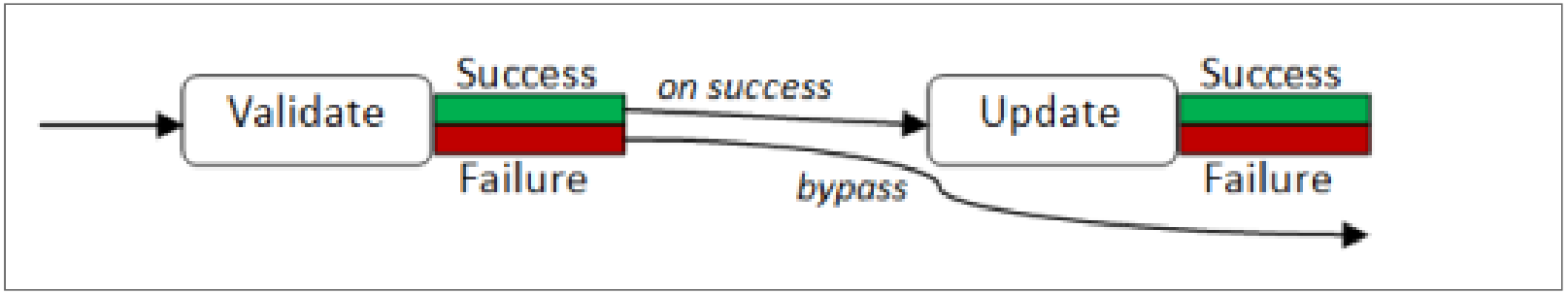
```
[HttpPost]
public HttpResponseMessage CreateCustomer(string name, string billingInfo)
{
    try {
        Customer customer = new Customer(name);
    } catch (Exception exc) {
        _logger.Log("..."); return Error("...");
    }
    if (customer == null) {
        _logger.Log("..."); return Error("...");
    }

    try {
        _repository.Save(customer);
    } catch (Exception exc) {
        _logger.Log("..."); return Error("...");
    }
    try {
        _paymentGateway.ChargeCommission(billingInfo);
    } catch (Exception exc) {
        _logger.Log("..."); return Error("...");
    }
    try {
        _emailSender.SendGreetings(name);
    } catch (Exception exc) {
        _logger.Log("..."); return Error("...");
    }

    return new HttpResponseMessage(HttpStatusCode.OK);
}
```

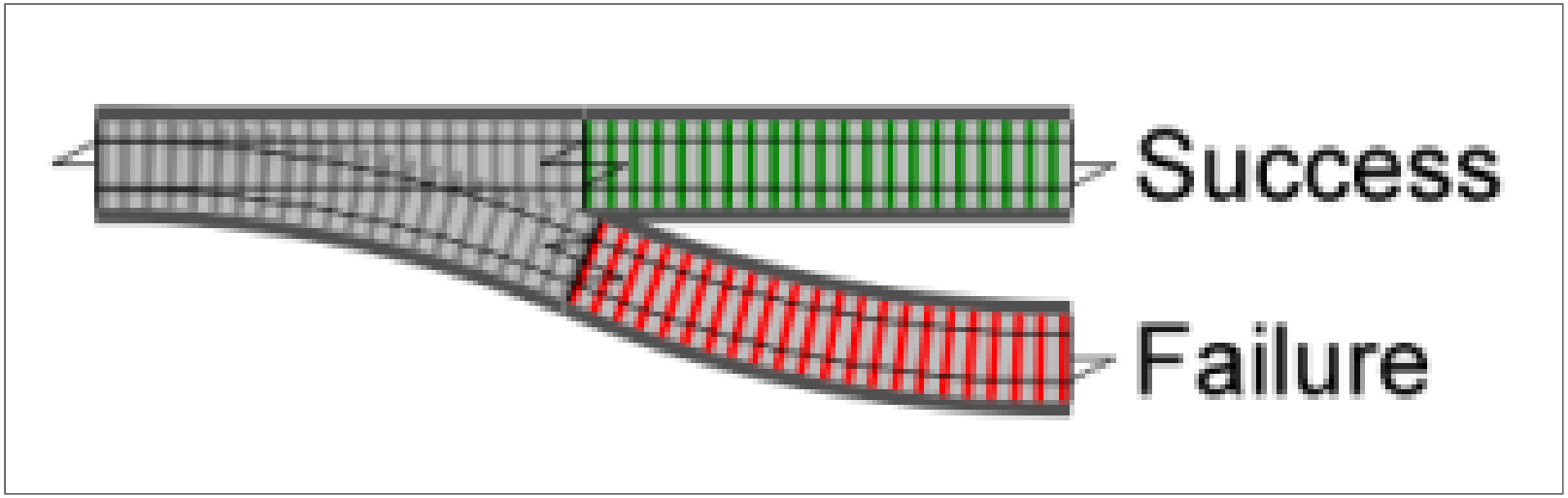
red·head·s
SOFTWARE SOLUTIONS SUPPORT





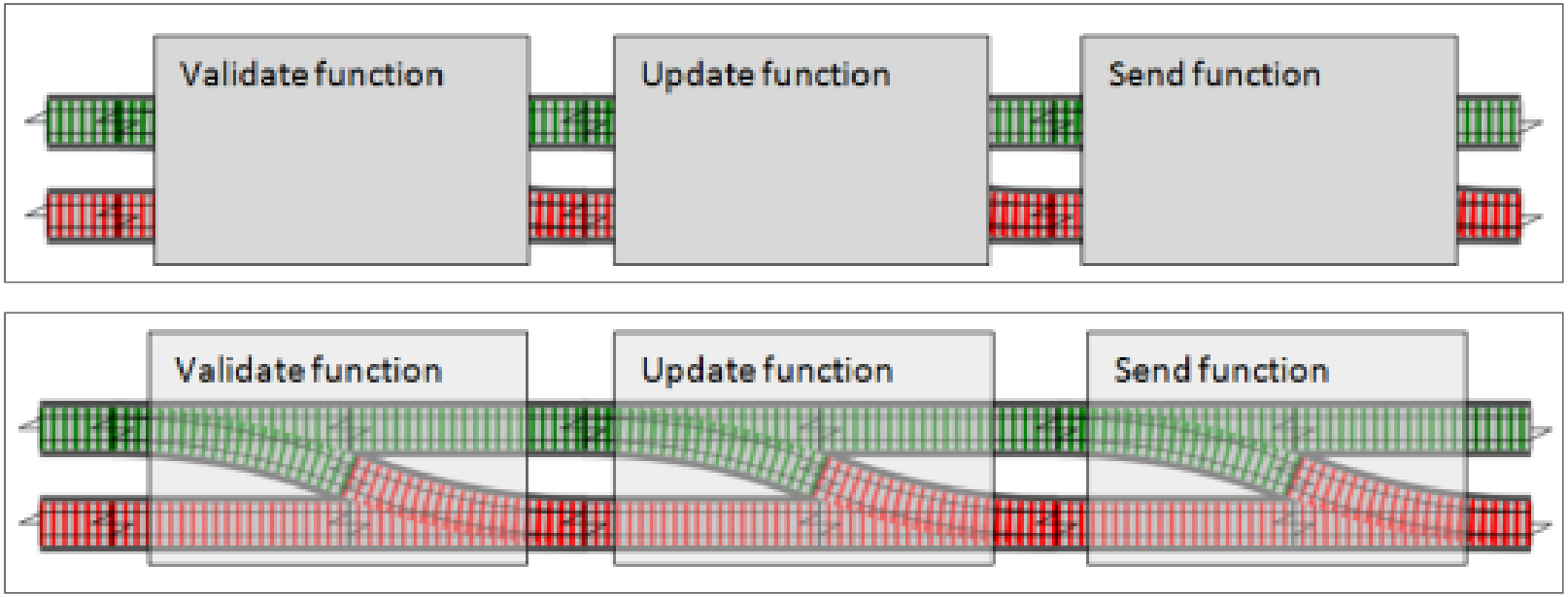
red·head·s
SOFTWARE SOLUTIONS SUPPORT

Wir haben 2 Ausgänge:



red·head·s
SOFTWARE SOLUTIONS SUPPORT

Wir brauchen **2 Eingänge**:



red·head·s
SOFTWARE SOLUTIONS SUPPORT

Für so eine Pipeline brauchen wir ein Objekt zum Durchreichen mit

- `Failure` und
- `Success<T>`

red·head·s
SOFTWARE SOLUTIONS SUPPORT

RESULT / MAYBE / OPTION

Generische Wrapper-Klasse(n),

- die `Success<T>` und `Failure` haben,
- die wunderbar erweiterbar sind

red·head·s
SOFTWARE SOLUTIONS SUPPORT

```
[HttpPost]
public HttpResponseMessage CreateCustomer(string name, string billingInfo)
{
    Result<BillingInfo> billingInfoRes = //..
    Result<CustomerName> customerNameResult = //..

    return Result.Combine(billingInfoRes, customerNameResult)
        .OnSuccess(() => /*...*/ )
        .OnSuccess(() => /*...*/ )
        .OnSuccess(
            /*...*/
                .OnFailure(() => /*...*/ )
        )
        .OnSuccess(() => /*...*/ )
        .OnBoth(result => /*...*/ )
        .OnBoth(result => /*...*/ );
}
```

red·head·s
SOFTWARE SOLUTIONS SUPPORT

```
[HttpPost]
public HttpResponseMessage CreateCustomer(string name, string billingInfo)
{
    Result<BillingInfo> billingInfoRes = //..
    Result<CustomerName> customerNameResult = //..

    return Result.Combine(billingInfoRes, customerNameResult)
        .OnSuccess(() => _paymentGateway.ChargeCommission(billingInfoRes.Value))
        .OnSuccess(() => new Customer(customerNameResult.Value))
        .OnSuccess(
            customer => _repository.Save(customer)
                .OnFailure(() => _paymentGateway.RollbackLastTransaction())
        )
        .OnSuccess(() => _emailSender.SendGreetings(customerNameResult.Value))
        .OnBoth(result => Log(result))
        .OnBoth(result => CreateResponseMessage(result));
}
```

LINKS

- **http://enterprisecraftsmanship.com/2015/03/20/functional-c-handling-failures-input-errors/**
- **http://fsharpforfunandprofit.com/rop/**
- **http://fsharpforfunandprofit.com/posts/recipe-part2/**

red·head·s
SOFTWARE SOLUTIONS SUPPORT