

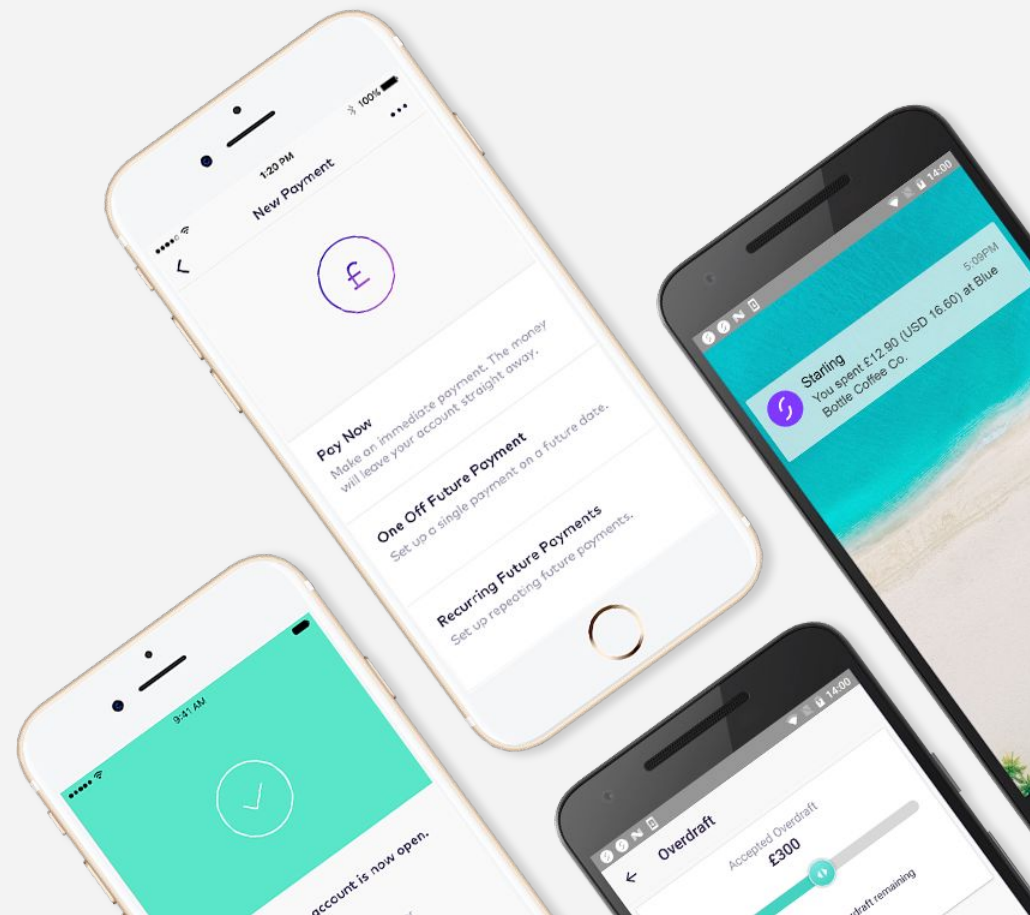
Rampant Pragmatism

Growth & Change at Starling Bank

Dan Osborne - Web Technology Practice Lead
Martin Dow - Engineering Lead for Core Banking



STARLING BANK



Starling Bank

- 2014 - Founded
- UK mobile retail bank.
- Joint accounts, sole traders, limited companies.
- Loans, Euro payments, foreign currency payments.
- Internet Bank
- 2019 - 1 million customers, £1 billion on deposit
- 2020 - Europe? 2+ million customers? ...

Agenda

- Complexity: Essence vs Accident
- A Relational Core
- The Rest of the System
- Our Web Stack
- Our Ledger
- Starling's Engineering Principles

Agenda

- Complexity: Essence vs Accident
- A Relational Core
- The Rest of the System
- Our Web Stack
- Our Ledger
- Starling's Engineering Principles

No Silver Bullet

Essence and Accident in Software Engineering

Frederick P. Brooks, Jr. - The Mythical Man-Month

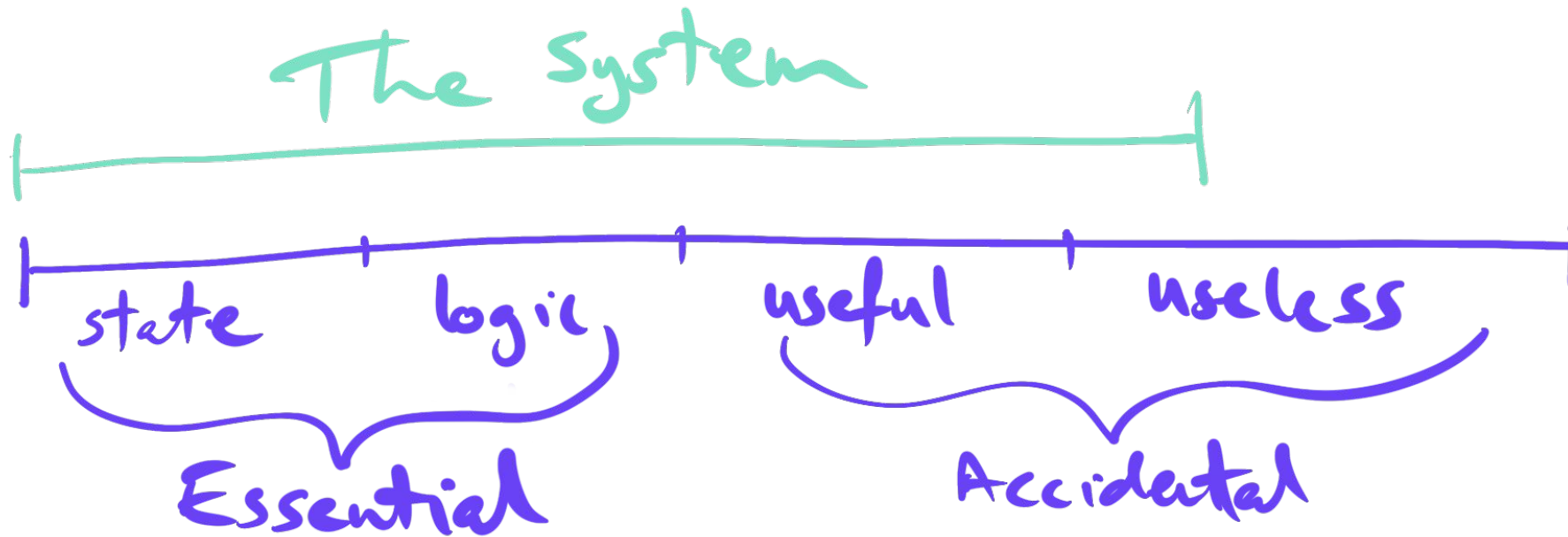
Essential Tasks

“The fashioning of the complex conceptual structures that compose the abstract software entity”

Accidental Tasks

“The representation of these abstract entities in programming languages and the mapping of these into machine languages within space and speed constraints.”

Spectrum of Complexity



Agenda

- Complexity: Essence vs Accident
- **A Relational Core**
- The Rest of the System
- Our Web Stack
- Our Ledger
- Starling's Engineering Principles

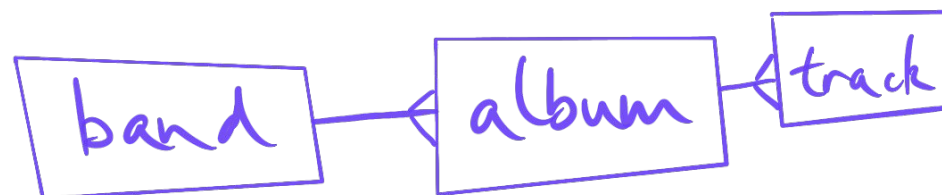
“I don’t care how much you really love the syntax of your favourite programming language, it’s inferior to data in every way”

Rich Hickey (referencing Gerald Sussman)

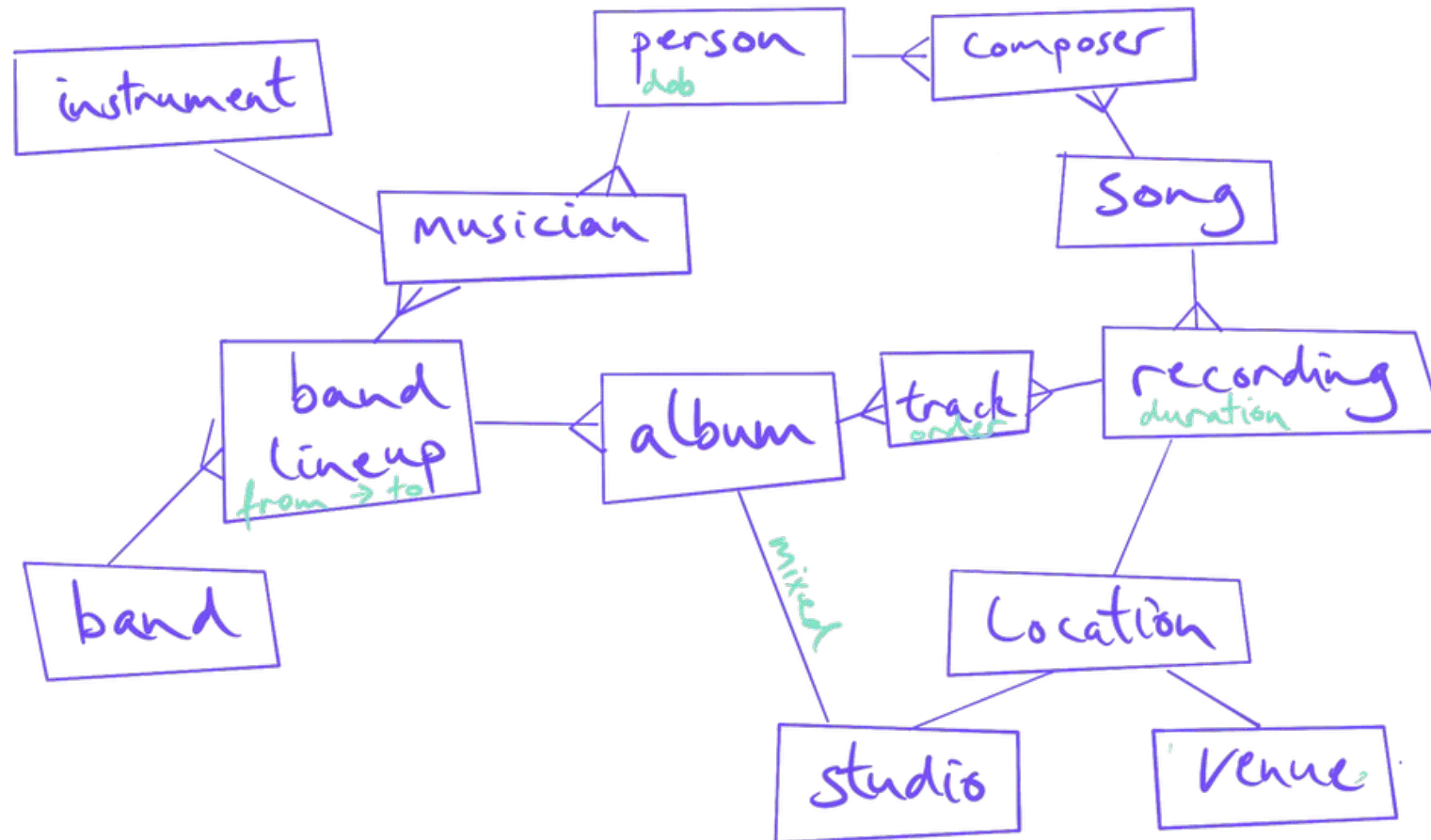
Essential Complexity

Data Modelling
as
System Design

Essential Complexity & Data Modelling



Flex the Model



Relational Modelling

Access Path Independence

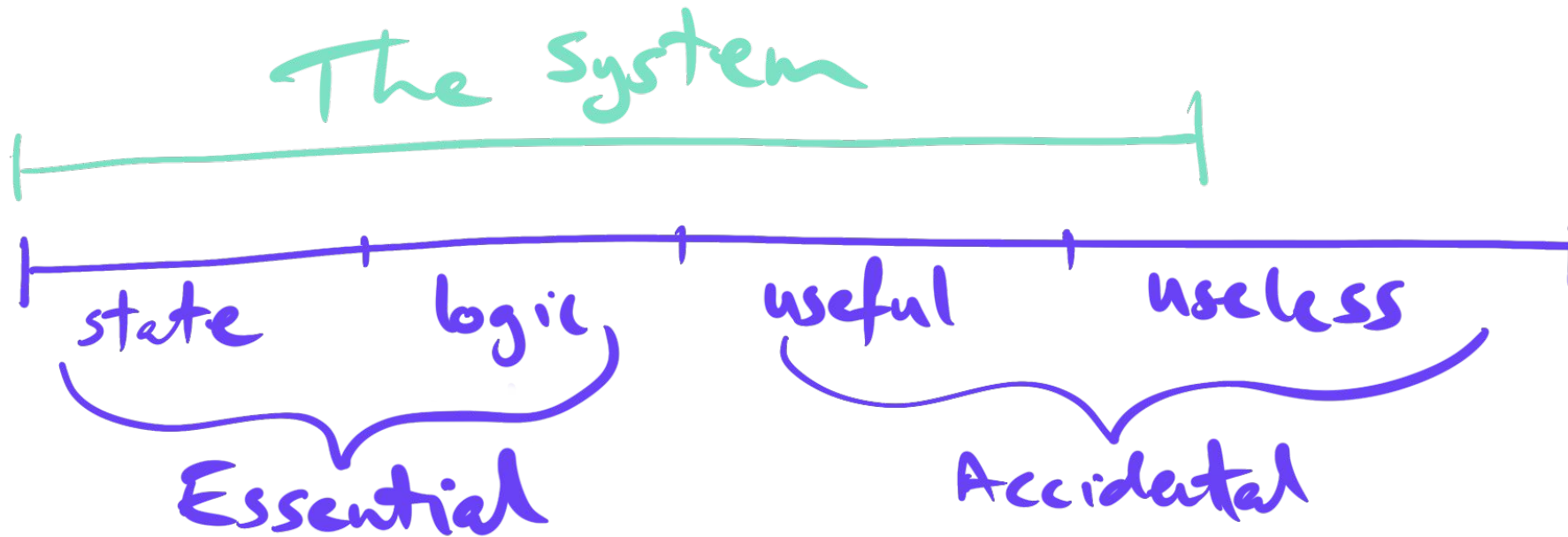
Postgres

- SQL: Access path independence
- Constraints: A declarative barrier

Agenda

- Complexity: Essence vs Accident
- A Relational Core
- The Rest of the System
- Our Web Stack
- Our Ledger
- Starling's Engineering Principles

Spectrum of Complexity



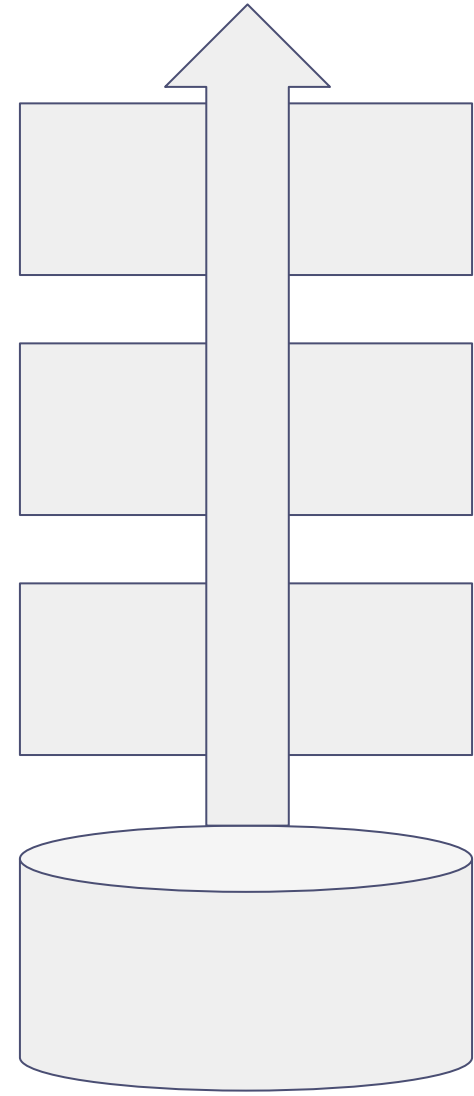
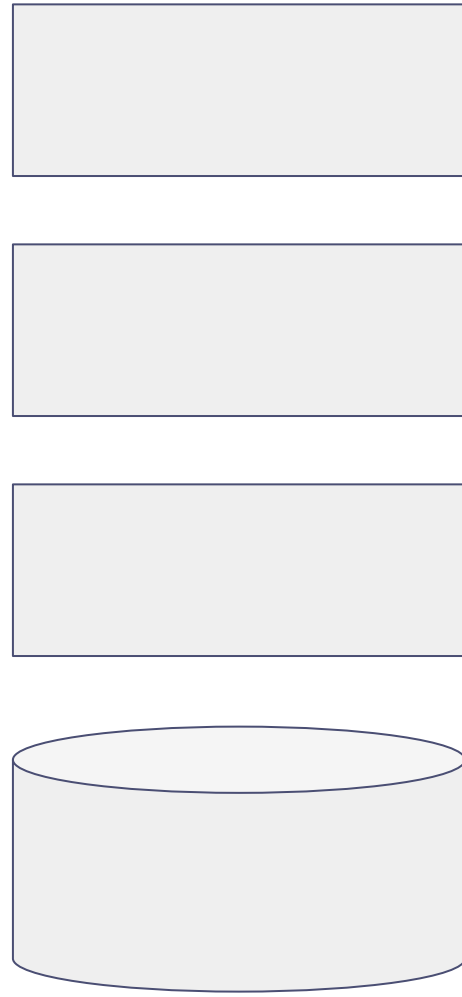
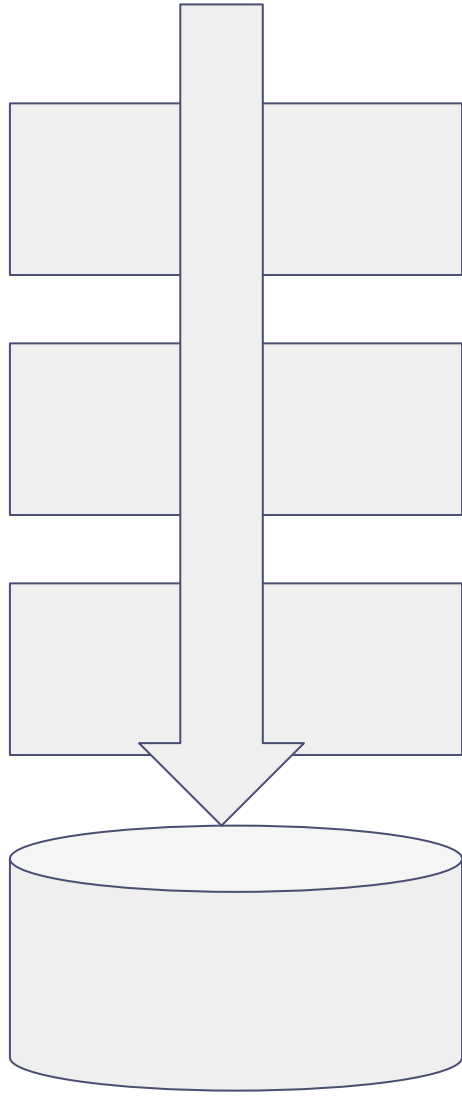
Not a System Yet

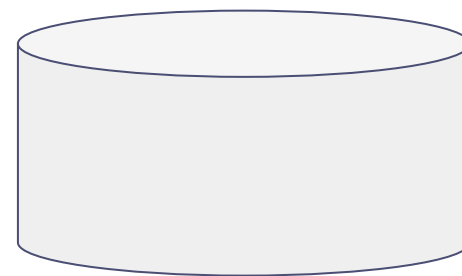
Out of the Tar Pit

What not How

But hang on...

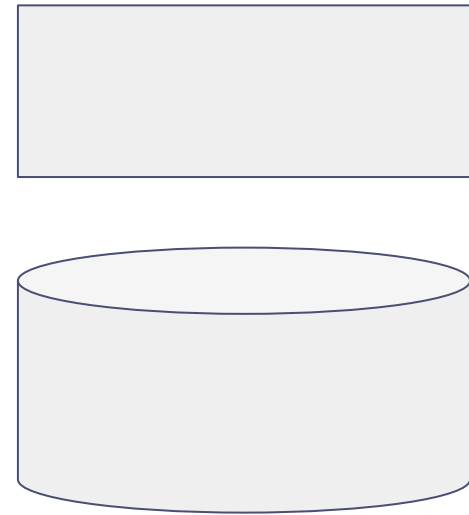
... you use Java?!





```
@GeneratedPersistence
@ImplementedBy(SprocketStatementsImpl.class)
public interface SprocketStatements {

    @DynamicQuery("select * from sprocket")
    Stream<SprocketRow> listAllSprockets();
}
```



```

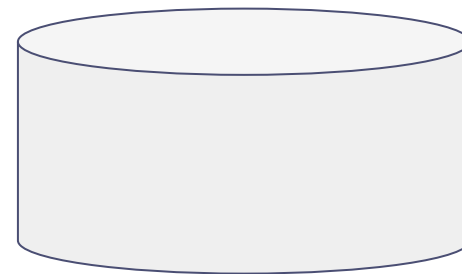
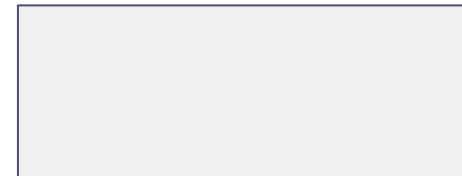
public class SprocketService implements SprocketResource {

    private final PersistenceContext persistenceContext;
    private final SprocketStatements statements;
    private final SprocketRowTransformer sprocketRowTransformer;
    private final AvailabilityFilter availabilityFilter;

    @Inject
    public SprocketService(
        PersistenceContext persistenceContext,
        SprocketStatements statements) {
        this.persistenceContext = persistenceContext;
        this.statements = statements;
        sprocketRowTransformer = new SprocketRowTransformer();
        availabilityFilter = new AvailabilityFilter();
    }

    public List<Sprocket> getAvailableSprockets() {
        return persistenceContext.inTransaction(() ->
            statements.listAllSprockets()
                .map(sprocketRowTransformer)
                .filter(availabilityFilter)
                .collect(Collectors.toList())
        );
    }
}

```

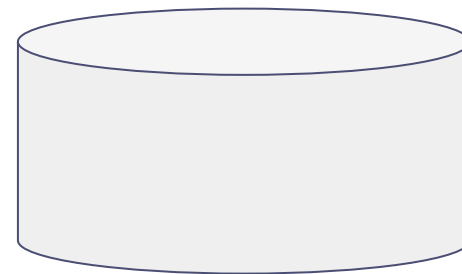
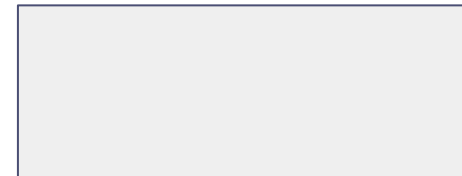
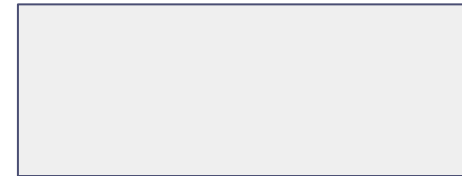



```

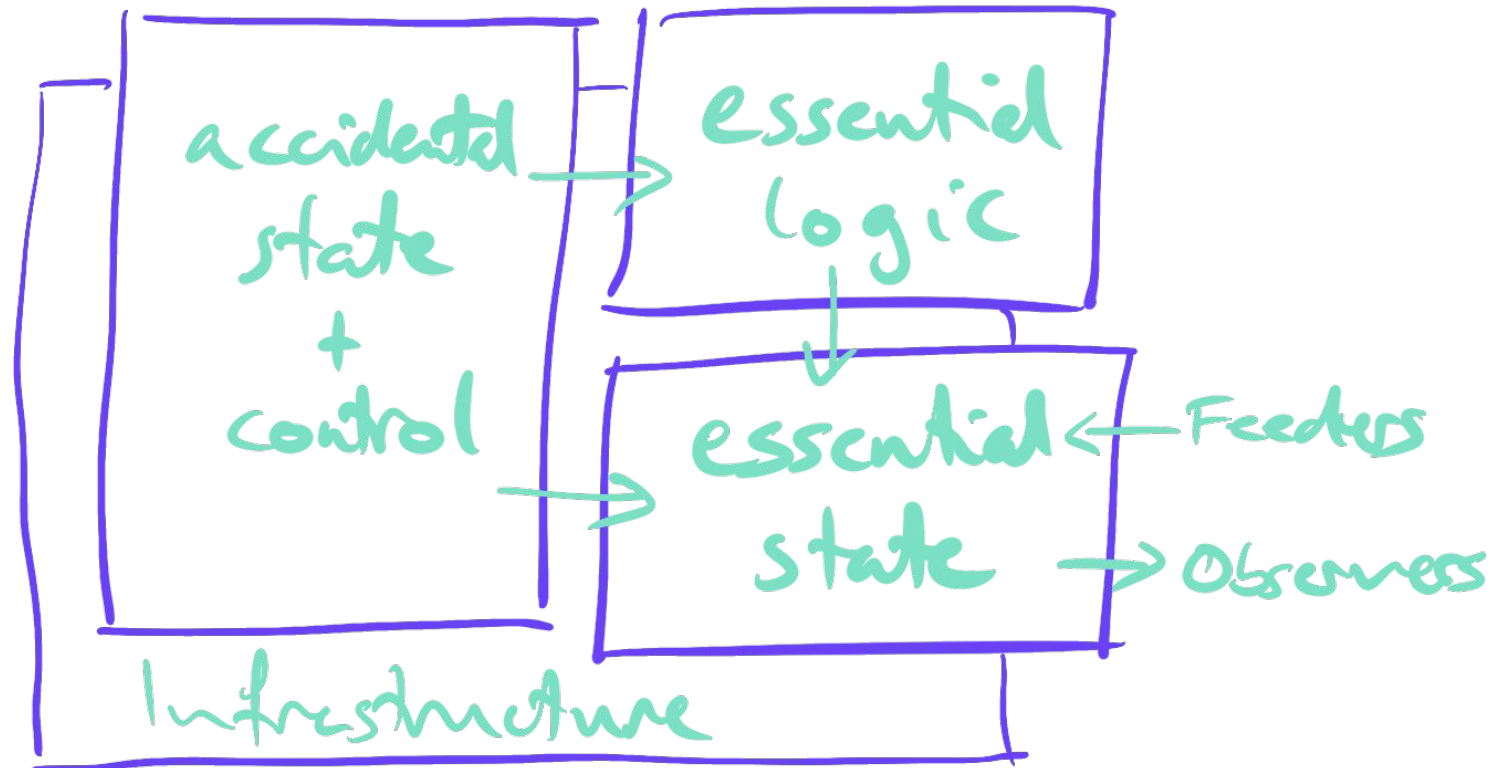
@Path("api/v2/sprockets")
@Api(value = "Sprocket API")
@Tag(name = "Sprocket API")
public interface SprocketResource {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    @Auth(type = AuthType.OAuth, scopes = Scope.AUTHORISING_INDIVIDUAL_READ)
    @ApiOperation(value = "Listing of available sprockets",
        nickname = "getAvailableSprockets",
        authorizations = @Authorization(
            value = "oauth2",
            scopes = @AuthorizationScope(
                scope = ScopeValue.SPROCKETS_READ,
                description = "")))
    @ApiResponses({
        @ApiResponse(code = 200, message = "Successful operation",
            response = Individual.class),
        @ApiResponse(code = 400, message = "Bad request",
            response = ErrorResponse.class),
        @ApiResponse(code = 500, message = "Server error")
    })
    @Operation(
        security = @SecurityRequirement(
            name = "oauth2",
            scopes = ScopeValue.SPROCKETS_READ),
        summary = "The full listing of available sprockets",
        responses = @io.swagger.v3.oas.annotations.responses.ApiResponse(
            responseCode = "200",
            description = "Successful operation",
            content = @Content(
                schema = @Schema(implementation = Sprocket.class))))
    @PublicApiResponse
    List<Sprocket> getAvailableSprockets();
}

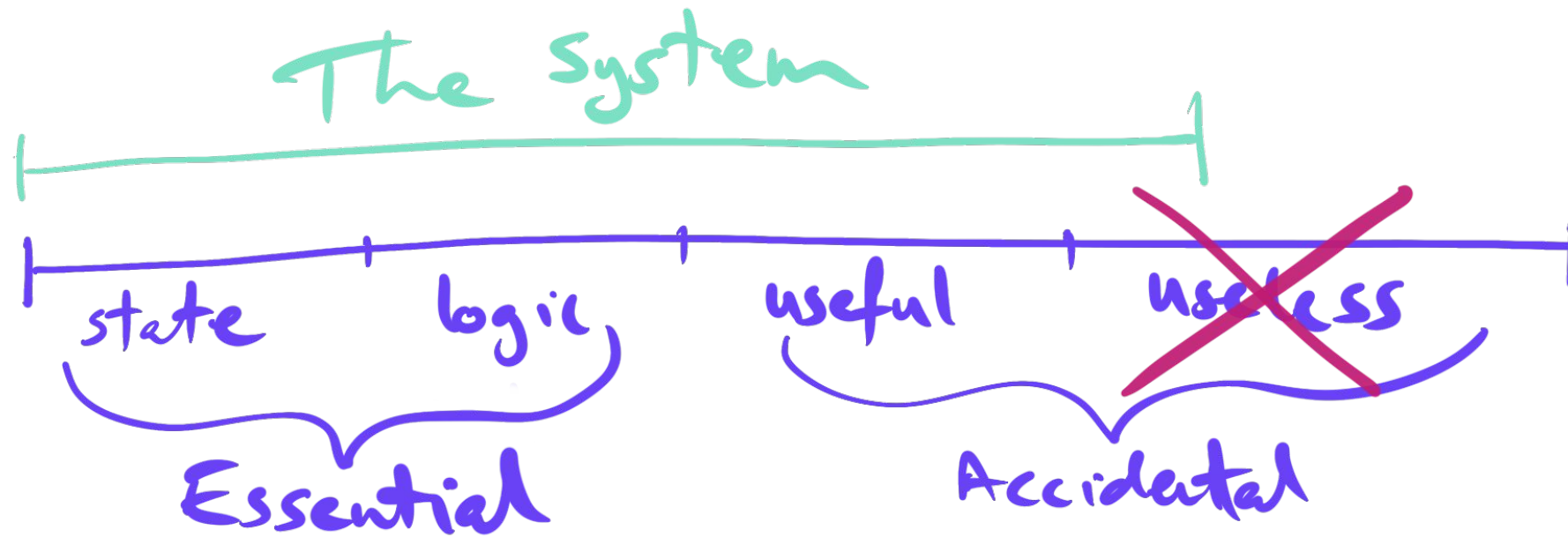
```



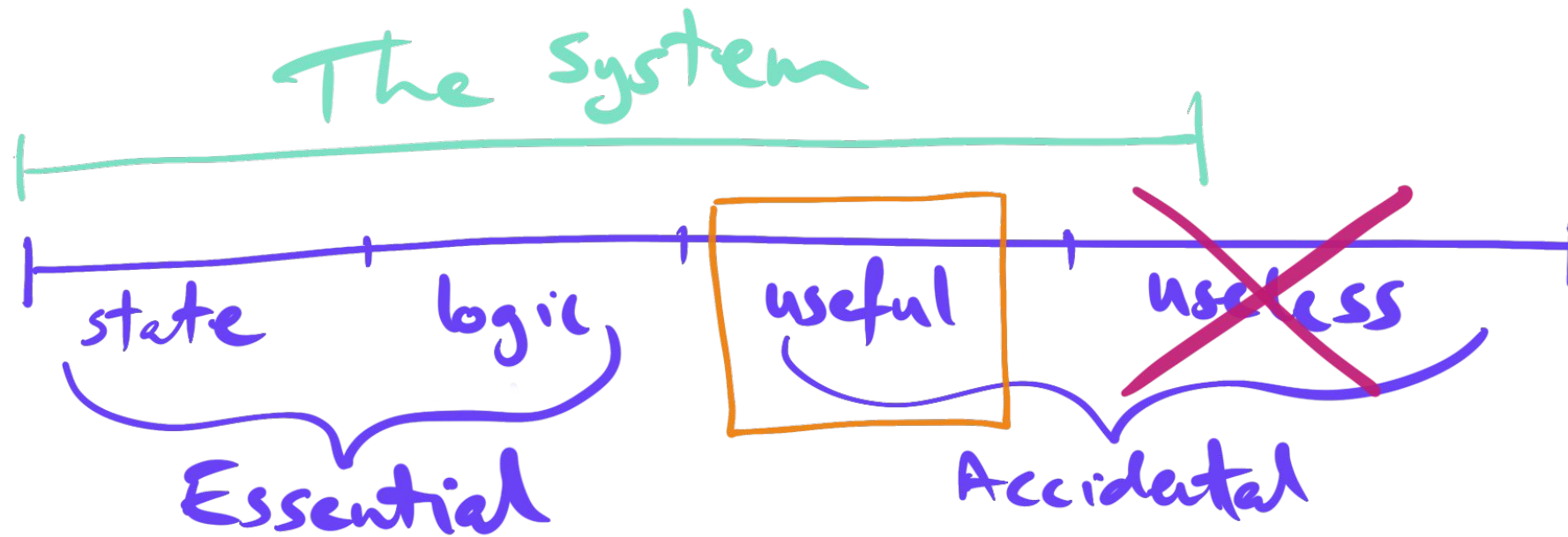
FRP System Components



- **Avoid**
- Separate



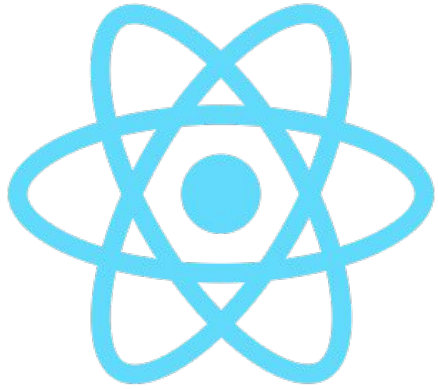
- Avoid
- **Separate**



Observers & Feeders

Agenda

- Complexity: Essence vs Accident
- A Relational Core
- Functional Relational Programming
- **Our Web Stack**
- Our Ledger
- Starling's Engineering Principles



React

```
import React, {useState} from 'react';

const gimmeTheTime = () => {date: new Date()};

export const SillyApp = () => <WhatTimeIsIt name='Dan Osborne' />;

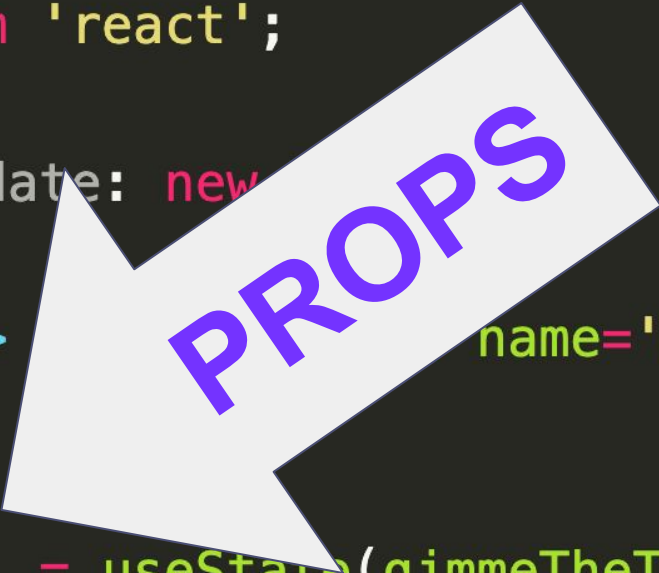
const WhatTimeIsIt = ({name}) => {
  const [theTime, setTheTime] = useState(gimmeTheTime());
  return <section>
    <h1>Hi, {this.props.name}!</h1>
    <button onClick={() => setTheTime(gimmeTheTime())}>
      What time is it now?
    </button>
    <h2>It is {theTime.toLocaleTimeString()}</h2>
  </section>;
};
```



```
import React, {useState} from 'react';

const gimmeTheTime = () => {date: new Date().toLocaleTimeString()};

export const SillyApp = () => {
  return (
    <div name='Dan Osborne' />
    <div>
      <h1>Hi, {this.props.name}!</h1>
      <button onClick={() => setTime(gimmeTheTime())}>
        What time is it now?
      </button>
      <h2>It is {theTime.toLocaleTimeString()}</h2>
    </div>
  );
};
```



```

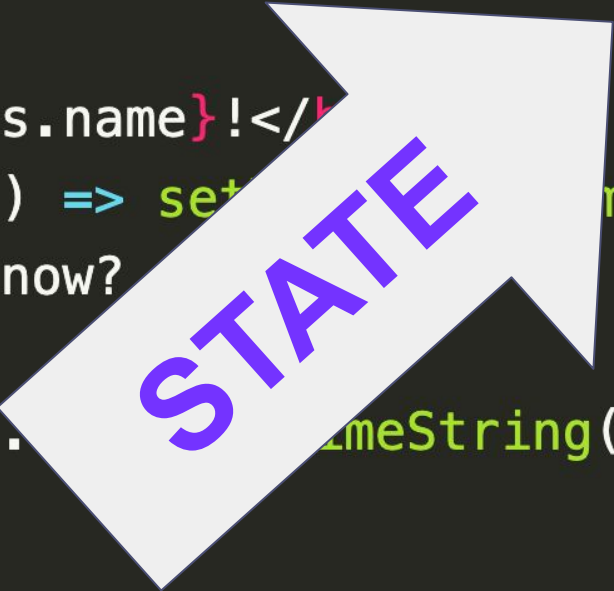
import React, {useState} from 'react';

const gimmeTheTime = () => {date: new Date()};

export const SillyApp = () => <WhatTimeIsIt name='Dan Osborne' />;

const WhatTimeIsIt = ({name}) => {
  const [theTime, setTheTime] = useState(gimmeTheTime());
  return <section>
    <h1>Hi, {this.props.name}!</h1>
    <button onClick={() => setTheTime(gimmeTheTime())}>
      What time is it now?
    </button>
    <h2>It is {theTime.toLocaleTimeString()}.</h2>
  </section>;
};

```



Functional Programs

“... provide a much clearer mapping between your ideas about how the program works and the code you actually write.”

Peter Seibel – Practical Common Lisp, 2005



Redux

```
const initialState = { sprockets: [] };
```

```
const ADD_SPROCKET_ACTION = 'ADD_SPROCKET';
```

```
const reducer = (state, {type, sprocket}) =>  
  type === ADD_SPROCKET_ACTION  
    ? Object.assign(  
      {...state},  
      {sprockets: [...state.sprockets, sprocket]})  
    : state;
```

```
const sprocketSelector = (state, sprocketId) =>  
  state?.sprockets?.find(id => id === sprocketId);
```

```
const SprocketAddingButton = (props) =>  
  <button onClick={props.dispatch({  
    type: ADD_SPROCKET_ACTION,  
    sprocket: {sprocketId: 42, name: 'Taper-Lock'}})}>  
    Add A Sprocket</button>;
```

```
const initialState = { sprockets: [] };
```

```
const ADD_SPROCKET_ACTION = 'ADD_SPROCKET';
```

```
const reducer = (state, {type, sprocket}) =>  
  type === ADD_SPROCKET_ACTION  
    ? Object.assign(  
      {...state},  
      {sprockets: [...state.sprockets, sprocket]})  
    : state;
```

```
const sprocketSelector = (state) =>  
  state?.sprockets?.find(sprocket => sprocket.name === name);
```

```
const SprocketAddingButton = (props) =>  
  <button onClick={props.dispatch({  
    type: ADD_SPROCKET_ACTION,  
    sprocket: {sprocketId: 42, name: 'Tape Sprocket'}  
  })}>  
    Add A Sprocket</button>;
```

CHANGE
STATE
HERE

Web Development

Accounts

People

Companies

Registrations

Onboarding

Duplicates

Cards

Faster Payments

Currency Cloud

Direct Debits

Stripe

CASS

Cheque

+ More

Daniel Osborne

Profile

Addresses

ID

Onboarding

Credit

Sanction

Duplicates

2LP

SARs

Terms

Notifications

Document Share

All Tabs

Person

View Details History

Scramble Mobile Number

Edit Profile

Merge Duplicate

Delete Person

Title

Mr

First Name

Daniel

Middle Name

TP

Last Name

Osborne

Date Of Birth

15/12/1979

Mobile Phone Number

+447941028060#456

Email Address

danosborne@example.com

App User

7612ebb7-24cb-4a1d-bb9d-809ee14f1a43

Address

45 Manor Street

South Exampleton

CR0 0AA

GB

Onboarding Status

Approved

Allowed in app

Marketing Consent

SMS

Email

Notification

Flags

Create Flag

History

Fscs Name Verified

01/03/2020 12:13:11

Verified 2020-01

Note

01/03/2020 12:13:25

Test Account

Owned Accounts

Individual

Daniel Osborne

Accessible Accounts

Link Account

ACCOUNT	AUTHORISED	REVOKED	UID
Daniel Osborne (GBP)	13/08/2019 20:09:31	<div>Revoke</div>	7a8bcf62-bdb6-497c-9b7e-c1a3c099ca4e
Daniel Osborne (EUR)	29/08/2019 17:38:14	<div>Revoke</div>	b8e57058-ffc-4a11-9df1-efe0b5a2f0f4

STARLING BANK

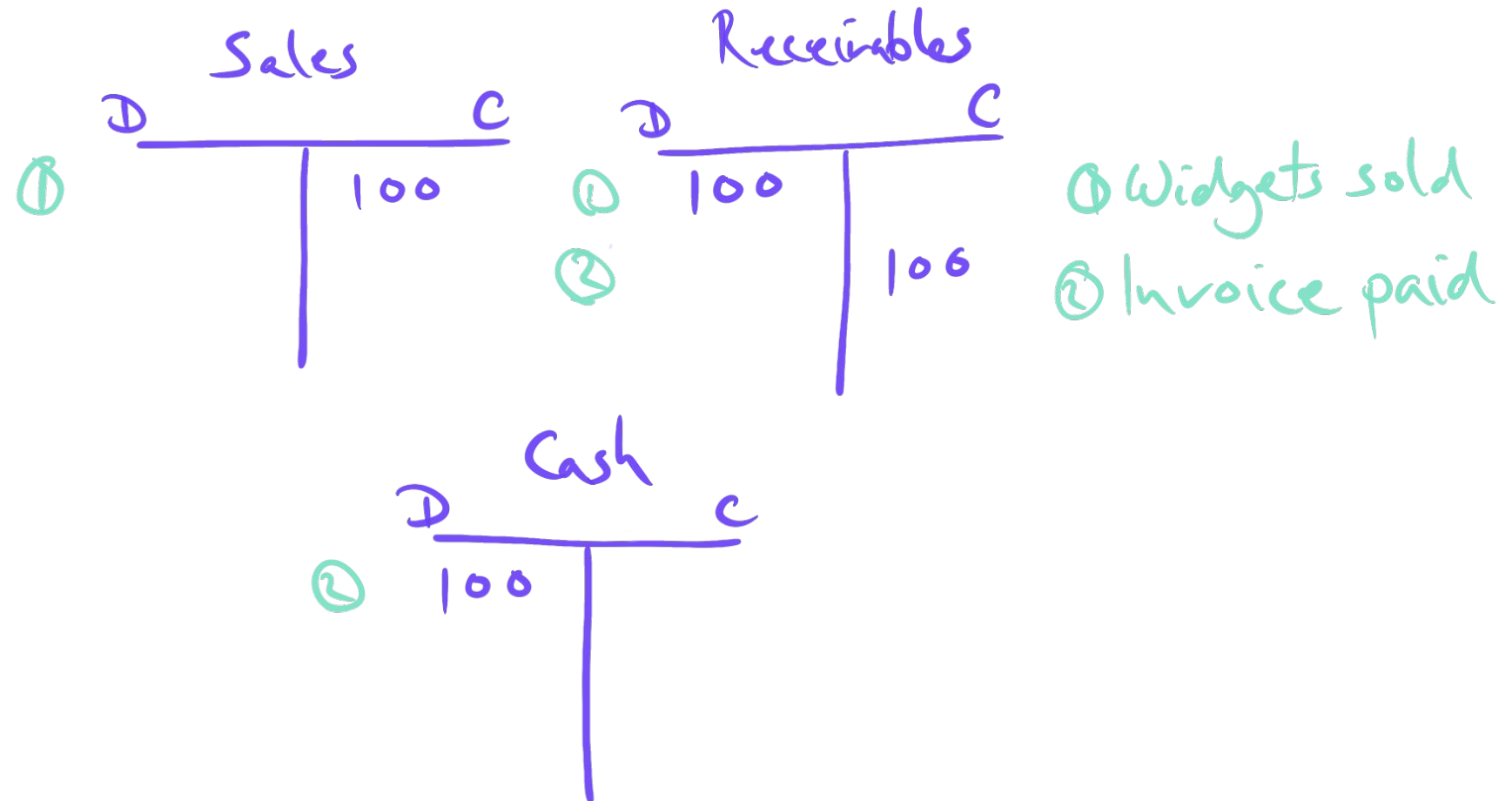
Web Development

... is FRP?!

Agenda

- Complexity: Essence vs Accident
- A Relational Core
- Functional Relational Programming
- Our Web Stack
- Our Ledger
- Starling's Engineering Principles

A Bank is an Accounting Machine



A Bank is an Accounting Machine

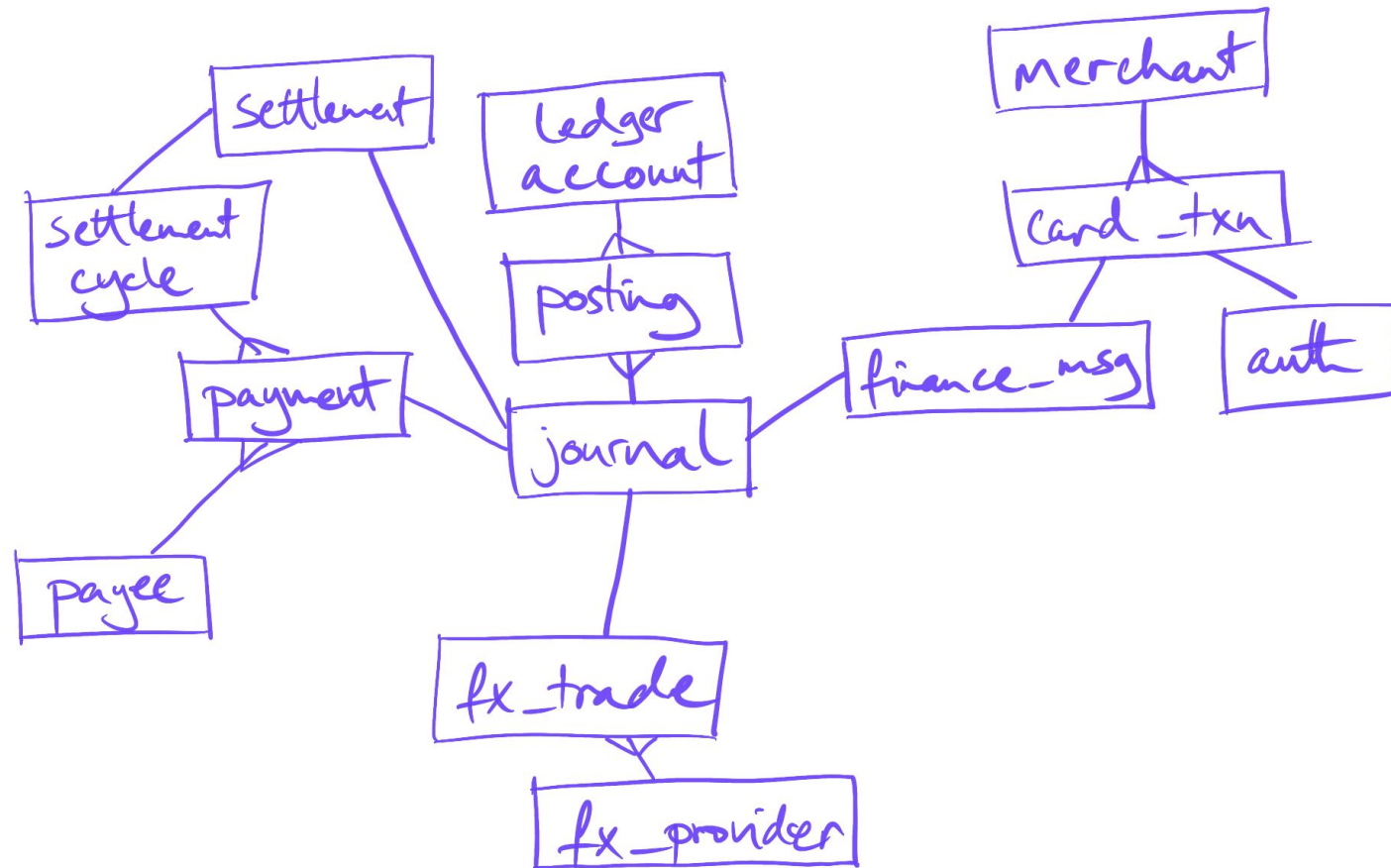
Detailed & Diverse

to

Unified & Generic

Denormalisation & Projection of Essential Data

Denormalised Postings

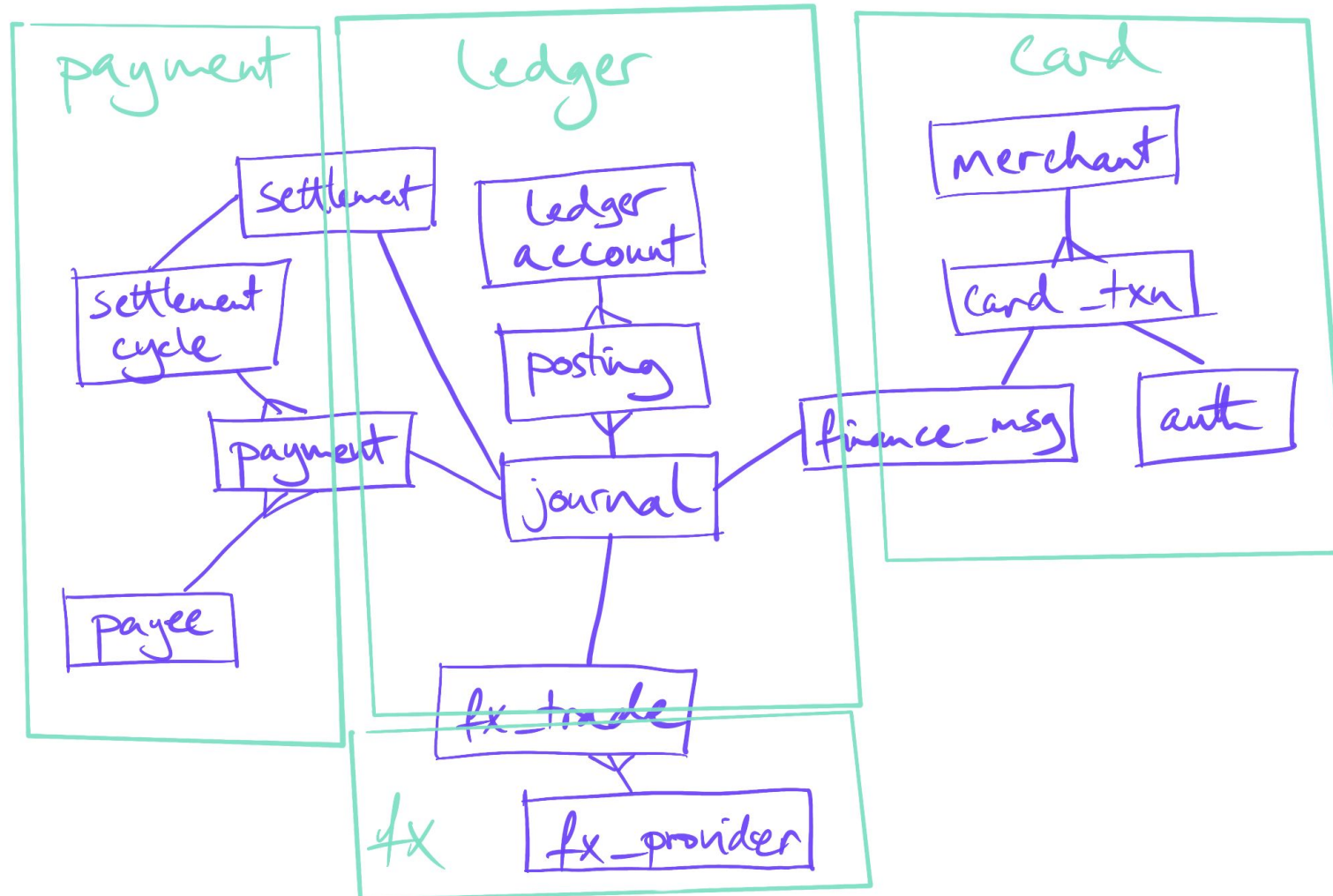


Separate Services?

Self Contained Systems

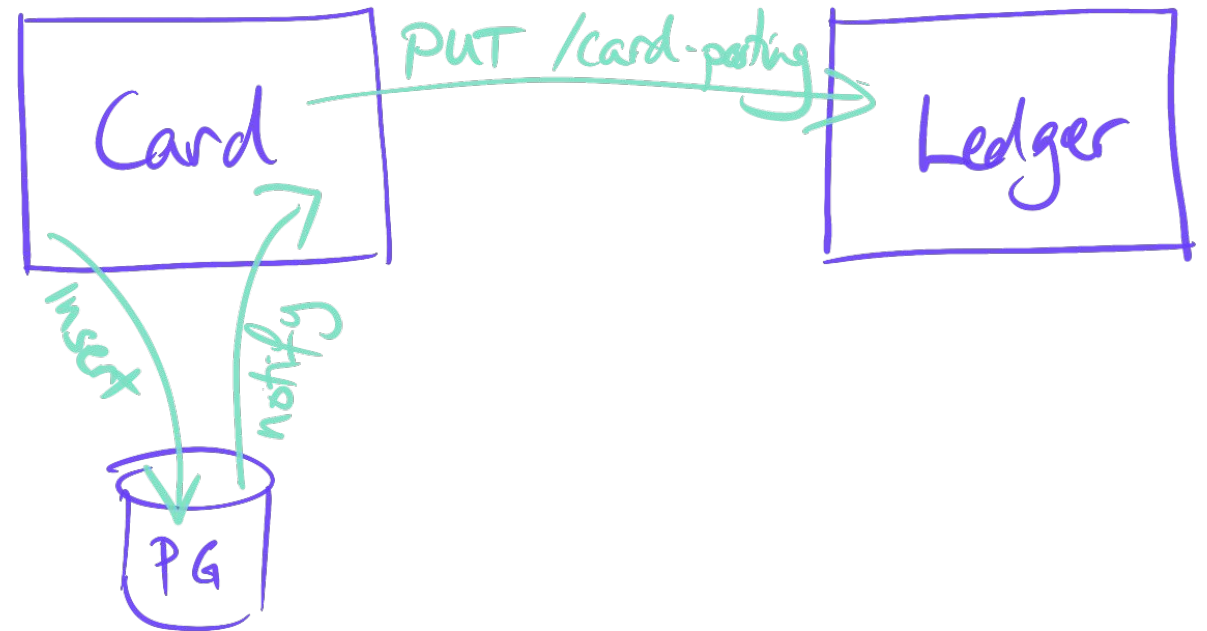
- Reduce blast radius
- Avoid distributed monolith
- Minimise synchronous calls
- Each service has its own database

Splitting the (Relational) Core



Pushing Data Around

- Each system as event source
- Database queues



Pushing Data Around

- Kafka?
- Postgres Logical Replication?

Agenda

- Complexity: Essence vs Accident
- A Relational Core
- Functional Relational Programming
- Our Web Stack
- Our Ledger
- Starling's Engineering Principles

Software Engineering

The Art of Compromise

Pragmatism

Moving Deliberately

Bets



Bets



Bets + **Principles** = Architecture

Bets + **Principles** = Architecture

Optimise
&
Empower

Bets + **Principles** = Architecture

Understandability

Bets + **Principles** = Architecture

Simplicity
&
Consistency

Rampant Pragmatism

Change & Growth at Starling Bank



Yodit Stanton

@yoditstanton



Micro services are dead, long live the monolith.....Next yr repeat the opposite. Or perhaps one day we will have a nuanced conversations on how to make appropriate tech choices. Dunno 🙄 perhaps a crazy idea

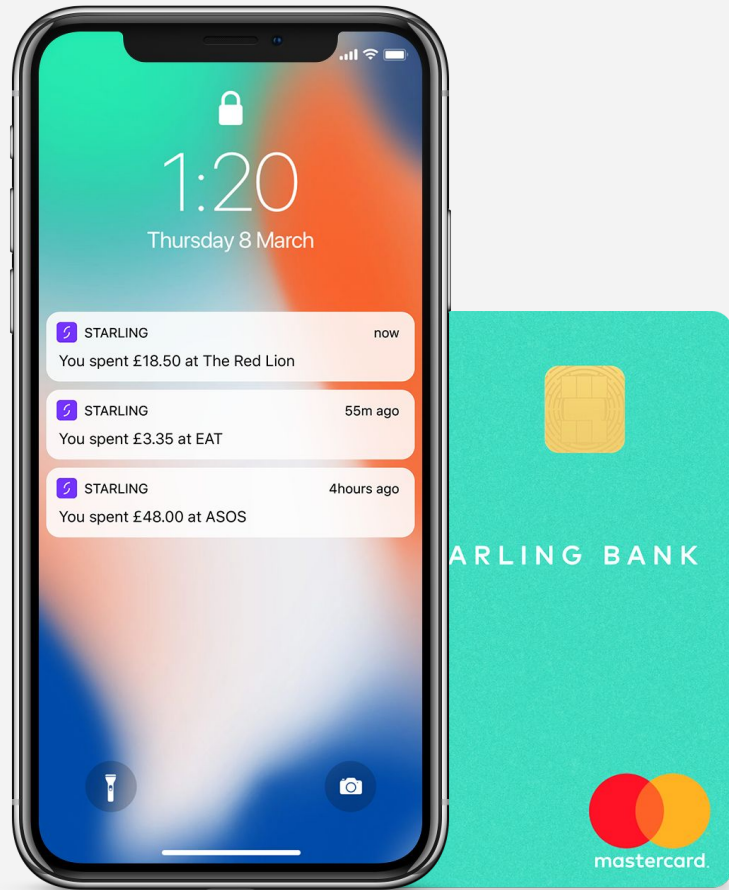
6:43 pm · 2 Feb 2020 · [Twitter for iPhone](#)

“Most startups (and big companies) don’t need the tech stack they have.”

Vicki Boykis

“You don’t need Kafka. Really.”

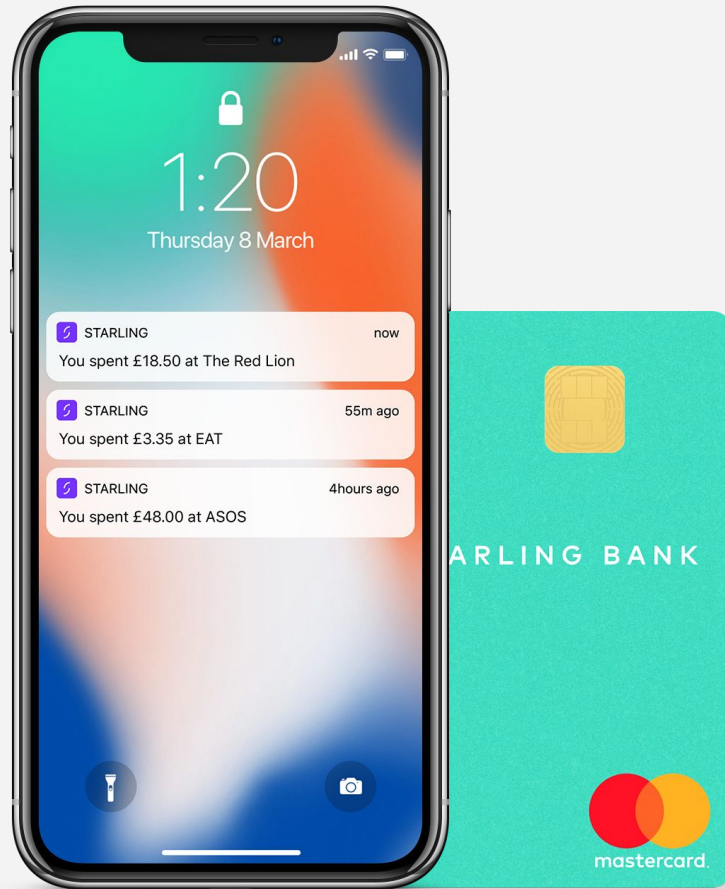
Normcore Tech Newsletter



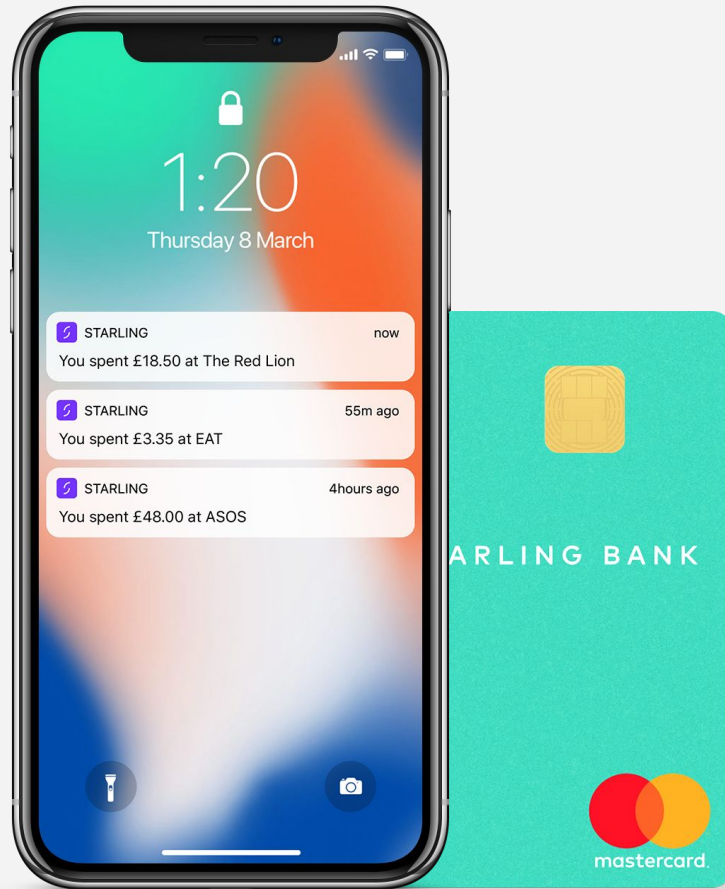
Thank you!

<https://developer.starlingbank.com>
<https://www.starlingbank.com/careers/engineering/>

@dtpo
@martin_dow
@StarlingDev



- No Silver Bullet - https://www.researchgate.net/publication/220477127_No_Silver_Bullet_Essence_and_Accidents_of_Software_Engineering
- A Relational Model for Large Shared Data Banks - <https://www.semanticscholar.org/paper/A-Relational-Model-for-Large-Shared-Data-Banks-Codd/c4bd2f89039031f09b9ddec07e6d456b0d08aab4>
- Out of the Tar Pit - <https://www.semanticscholar.org/paper/Out-of-the-Tar-Pit-Moseley-Marks/41dc590506528e9f9d7650c235b718014836a39d>
- Simple Made Easy - <https://www.infoq.com/presentations/Simple-Made-Easy-QCon-London-2012/>



- Object Oriented Software Construction - <https://www.semanticscholar.org/paper/Object-Oriented-Software-Construction-Meyer/5f0e007b600d595b9c75cf3949d29b6ae21eed63>
- Mythical Man Month - <https://www.bookdepository.com/Mythical-Man-Month-Frederick-P-Brooks-Jr/9780201835953>
- Practical Common Lisp - <https://www.bookdepository.com/Practical-Common-Lisp-Peter-Seibel/9781430211617>
- Applied Mathematics for Database Professionals <https://www.apress.com/gp/book/9781590597453>