Rampant Pragmatism
Growth & Change at Starling Bank

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

The System

state       logic       useful       useless
Essential   Accidental
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

The System

- State
- Logic

Essential

- Useful
- Useless

Accidental
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 = "")
        ))
    @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

export const SillyApp = () => 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()};

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.toLocaleString()}.</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
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?.sprockets?.find(id => id === props.id);

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

Sales

\[ \begin{align*}
\text{D} & \quad \text{100} \\
\end{align*} \]

Receivables

\[ \begin{align*}
\text{D} & \quad \text{C} \\
\text{100} & \quad \text{106} \\
\end{align*} \]

Cash

\[ \begin{align*}
\text{D} & \quad \text{C} \\
\text{100} & \\
\end{align*} \]

@ Widgets sold
@ Invoice paid
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
Micro services are dead, long live the monolith.....Next yr repeat the opposite. Or perhaps one day we will have a nuisanced 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/2204771
  27_No_Silver_Bullet_Essence_and_Accidents_of_So
  ftware_Engineering

- A Relational Model for Large Shared Data Banks -
  https://www.semanticscholar.org/paper/A-Relatio
  nal-Model-for-Large-Shared-Data-Banks-Codd/c4bd
  2f89039031f09b9ddec07e6d456b0d08aab4

- Out of the Tar Pit -
  https://www.semanticscholar.org/paper/Out-of-the
  -Tar-Pit-Moseley-Marks/41dc590506528e9f9d7650c
  235b718014836a39d

- Simple Made Easy -
  https://www.infoq.com/presentations/Simple-Made
  -Easy-OCon-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