Never again offline?!?
Experiences on the outstanding role of data in a large-scale mobile app ecosystem

Dr. Matthias Naab, Fraunhofer IESE
Dr. Ralf Carbon, John Deere ETIC
Susanne Braun, Fraunhofer IESE
John Deere Ecosystem
Introducing Mobile to the John Deere Ecosystem

- One app vs. many apps
- Native vs. hybrid or web
- Online vs. Offline
- Proprietary vs. standard data
- ...

My John Deere
Collaboration Project

- Duration: since Q4 2013
  - Several phases and scope extensions
- Team
  - ~ 6 full time persons
- Approach
  - UX / Arch / Dev / QA
  - 2 week iterations
  - Testing with real end customers since summer 2014
  - Continuously feeding concepts into production version
  - Re-newing the code base after ~6 months
Where to find Data in a Typical Architecture Document?

[Diagram showing UI, Logic, Data with an arrow pointing to Data and the word “Here!”]
Where is Data Really???

UI

Logic

Data

Everywhere!
What does this mean for architecting?
We need an App ... Architecting it should be Simple!
We need an App … Architecting it should be Simple!

JobManager iOS App
- UI
- CoreData
- DB: SQLite

iOS Data Modeling

© Fraunhofer IESE
Multiple Users Work on Shared Enterprise Data
Multiple Users Work on Shared Enterprise Data

- Data exchange via backend

Device 1: iPad
  - JobManager iOS App

Device 2: iPhone
  - MyJobs iOS App

Cloud Backend
  - Mobile Services
  - Logic
  - Hibernate
  - DB: MySQL

HTTP
REST
JSON
Apps Work on Shared Data
Apps Work on Shared Data

- iOS <=7: No shared data
Apps Work on Shared Data

- Data exchange via backend
- On app switch, update data
Users Have to Work Offline
Users Have to Work Offline

- Custom sync mechanism
- Sync service and engine
- UUID + RevisionID + SyncState
- Conflict detection + resolution
- Treatment of deletion
- \( \rightarrow \) High effort!

```json
job {
    "id": "UUID",
    "revNr": "42",
    "syncState": "UPDATED"
}
```
Data has Different Characteristics

- Data classification
- Data modeling rules (e.g. transactional accesses master, usage of aggregates)
- Life-cycle of objects (deletion)
JD Operates Multiple Tenants on the Backend

- Separation of tenants with tenantId for each object
- General data provided by JD
- Sync has to respect tenants
- Cost-efficient operation
Multiple Users Share one Device

iOS apps and devices not prepared for multi-user
Multiple Users Share one Device

- Login mechanism for users
- Custom realization of SSO
- Usage of keychain group access for SSO tokens
- Sharing among apps via backend
- Removing data of logged out users from current SQLite DB
Connecting to the World

- Collecting data on the field
- Streaming map data
- Reusing weather data across tenants
- Sharing master data with other management systems
Increasing Amount of Data

- > 10,000 objects
- Continuously growing
- Offline requires data sync
- Slow initial sync (> 60s)

- Filter old transactional data
- Compress
- Prefill SQLite DB file
- Deeply optimize slow CoreData storage
- ...

Device 1: iPad
- JobManager iOS App
  - UI
  - Sync Engine
  - CoreData
  - DB: SQLite

Cloud Backend
- Sync
- Mobile Services
  - Logic
  - Hibernate
  - DB: MySQL
Increasing Frequency of Changes of Data

- CQRS: Separate interface for high frequency data
- Event-Sourcing
- Cassandra for fast writes
- In-memory for fast reads

Near real-time data (GPS, ...) → ~1 change/s
Live distribution to devices
Visualization and analysis

Device 1: iPhone

Logistics iOS App

Cloud Backend

- HFR
- HFW
- Sync
- Mob. Serv.
- In Memory
- Cassandra

DB: MySQL

Logic
Hibernate
Upgrading Apps

- Support multiple app versions
- Maintain single DB model
- Create mapper
Multiple Languages to be Supported

Language resources for texts

Translated master, reference, and transactional data

- Metrics / units
- Where to transform units?
- In which units to persist?
- Tenant-spanning analyses

Cloud Backend
- Sync
- Mobile Services
- Logic
- Hibernate
- DB: MySQL

Device 1: iPad
- JobManager iOS App
- UI
  - English

Device 2: iPad
- JobManager iOS App
- UI
  - German
Summary: Overview of Data Representations

Cloud Backend

- HFR
- HFW
- Sync
- Mob. Serv.
- Logic
- Hibernate
- DB: MySQL

Device 1: iPad

- JobManager iOS App
  - UI
  - Sync Engine
  - CoreData
  - DB: SQLite

Java Objects

Cassandra Column F.

CoreData Objects

Relational Tables

JSON DTOs

Java DTOs

Java Entities

Relational Tables
Lessons Learned – Impact on Quality Attributes

UX and performance:
high impact on data architecture

Security:
makes offline capability even more difficult

Maintainability:
nearly always adversely impacted by concepts for other quality attributes
Lessons Learned – Technical Aspects

- Offline capability: not out of the box
- Offline capability: costly to develop
- CoreData: requires in-depth technical know-how
Lessons Learned – Data Modeling

There is not THE single data model

Data modeling is highly intertwined with architecture design

Data modeling according to OOD / DDD is not enough
Data is often neglected in architecture
Data is highly related to quality attributes
Design and document decisions around data!
Farming is innovative and strongly uses IT 😊
Dr. Matthias Naab
Phone: +49 631 6800 2249
matthias.naab@iese.fraunhofer.de

Dr. Ralf Carbon
Phone: +49 631 3619 1813
carbonralf@johndeere.com
Appendix: Legend of Diagrams

- **Component**
- **App**
- **Data object**
- **Computational Node**
- **Connector**

Yellow: Introduced or changed here