Platform Architecture for OmniChannel Retail

ProductDNA
June 21, 2018
Background

VP, Architecture and Chief Architect
Plan for Today

- Context of the omnichannel retail environment
- Legacy architecture overview
- Platform architecture
- Platform examples
- Platform framework
- Platform learnings
Target is Omnichannel

• 1,829 stores in the United States
• 39 distribution centers in the U.S
• 350,000+ team members worldwide
• Global locations in India
• Target.com is the fourth most-visited retail website in the U.S. with more than 26 million unique visitors each month on average
75% of Americans live within 10 miles of a Target store.
Over 60% of all digital sales are fulfilled by a store.
# Omnichannel Retail

<table>
<thead>
<tr>
<th>Commerce</th>
<th>Customer Interaction</th>
<th>Customer Fulfillment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>Store Associate</td>
<td>Shopping bags</td>
</tr>
<tr>
<td>Online</td>
<td>Call Center</td>
<td>Ship to home</td>
</tr>
<tr>
<td>Mobile</td>
<td>Mobile App</td>
<td>Ship to store</td>
</tr>
<tr>
<td>Partner</td>
<td>Online Chat</td>
<td>In-Store pickup</td>
</tr>
<tr>
<td>Voice</td>
<td>Email</td>
<td>Car park pickup</td>
</tr>
<tr>
<td>Text</td>
<td>Text</td>
<td>Same day delivery</td>
</tr>
<tr>
<td>Social</td>
<td>Social</td>
<td>Partner direct ship</td>
</tr>
<tr>
<td></td>
<td>Augmented Reality</td>
<td>Digital goods</td>
</tr>
</tbody>
</table>
Omnichannel Retail Evolving Rapidly

Every step of the retail value chain is being disrupted
• Import and Logistics (Direct to Consumer, Crowdsourced Delivery)
• Selection & Curation (Online Search and Recommendations)
• Trips to Stores (Same Day Delivery, Personal Shoppers)
• Online Mega-Retailers (Everything Store)
• Online Micro-Retailers (Single Category Specialists)
• Checkout process (Self-checkout, Automated checkout)
Legacy Architecture
Legacy Footprint

70% of capabilities tied to mainframe

7000+ RDBMS

Operationally siloed

3000+ Applications

Stores

Digital

Supply Chain

Marketing

Merchandising

Product Development

Corporate
Each box contains infrastructure onsite
Legacy Architecture
Mainframe Dependency Graph

Each node is a table, each name is an application that writes to that table
What is a Platform?

A set of technologies that are the fundamental building blocks of custom applications

A Platform Has:

1. Primitive components
2. A defined surface
3. Extension points
What is a Retail Platform?

A set of primitive APIs and Services that represent the data, processes and business logic required to complete customer transactions.
Retail Platform Primitives - Examples

**Data**
- Item API
- Price API
- Inventory API
- Location API
- Tax API
- Customer API
- Worker API

**Process and Logic**
- Checkout API
- Cart API
- Restrictions API
- Returns API
- Address Verification API
- Item Movement API
- Shift Management API

A Platform Has: **Primitive components**
Retail Defined Surface

The complete set of retail platform primitives that define the core components of a retailer

A Platform Has: A defined surface
Retail Platform Extension Points

All retail platform primitives can be extended by users of the platform

A Platform Has:

Extension points
Fundamental Data

Data that **cannot** be derived from other data or is **generated** during common business processes of the company, divided into the logical domain entities of the business.

Retail Platform Architecture  

Primitive component
Fundamental Data Aggregations

Pre-joined and cached fundamental data, used to pre-calculate commonly used data patterns and protect fundamental data services from excessive load.

Retail Platform Architecture

Primitive component
Fundamental Business Process

The generic components of a business process, presented as an API, that can be used by all channels that execute the business process.
Fundamental Business Logic

The proprietary logic of the business, presented as an API, that is used by all channels that require that business logic.

Restrictions API  Cart Price API  Worker Pay API  Address Verification API

Retail Platform Architecture  Primitive component
Fundamental Platform Components

- Business Logic
  - Platform Logic Data
- Business Process
  - Platform Process Data
- Data Aggregation
  - Platform Cache

Primitive components:
- Price API
- Promo API
- Customer API
- Location API
- Tax API
- Order API
- Item API
- Worker API
- Vendor API

Retail Platform Architecture
Retail Platform Surface

Business Logic
- Restrictions API
- Cart Price API
- Worker Pay API
- Addr Verify API

Business Process
- Checkout API
- Cart API
- Payment API
- Item Setup API
- Fulfillment API
- Offer Setup API
- Returns API

Data Aggregations
- Item-Location-Price API
- Customer-Orders API
- Item-Offer API

Fundamental Data
- Price API
- Promo API
- Customer API
- Location API
- Tax API
- Order API
- Item API
- Worker API
- Vendor API

Retail Platform Architecture
A defined surface
What is a Tenant?

A user of the platform that builds applications using and extending the platform primitives

• Tenant drives an interaction with an actor
• Tenants are coarse grained around a channel (digital, store, supply chain, corporate)
• Isolation from other tenants
• Can only call services within the tenant, or provided by the platform
Single Tenant Components

- User Interface
- View Controller
- Business Logic Extension
- Business Process Extension
- Tenant Specific APIs
- Service Aggregator
- Tenant Logic Data
- Tenant Process Data
- Tenant Data
- Forward Cache

Retail Platform Architecture

Extension points
Retail Platform Architecture

Tenant and Platform

Tenant
- User Interfaces
- View Controllers
- Tenant Services
- Forward Cache

Tenant
- User Interfaces
- View Controllers
- Tenant Services
- Forward Cache

Tenant
- User Interfaces
- View Controllers
- Tenant Services
- Forward Cache

Business Logic APIs
Business Process APIs
Data Aggregation APIs

Fundamental Data APIs – READ ONLY

Price
Promo
Customer
Location
Tax
Order
Item
Worker
Vendor

Tenant and Platform
Eventual Consistency

• The Fundamental Data layer contains the true real-time operational data – source of truth
• Fundamental Data is exposed to the Platform as READ ONLY
• All other layers are caches
• Tenants operate almost exclusively off caches
• Tenant applications must be designed for eventual consistency
Events

• All data changes are events
• Tenants *do not* write to Fundamental Data
• Tenants *emit* events
• Fundamental Data *listeners* process events
Retail Platform Architecture
Retail Platform Architecture

Data Analytics

Merchandising Tenant
- Price Management
  - Item-Price Cache

Price Management Tenant Services
- Forward Cache

Tenant
- Tenant Services

Business Logic APIs
Business Process APIs
Item-Price API

Price API

Real Price Event

Price Event Listener

Price Event Listener

Price Data Set

Price

Price Update

Price Update
Process or Logic Extension
Retail Platform Architecture

Process Extension

1. List of items
2. Price and promos
3. Fulfillment options
4. Payment type
Retail Platform Architecture

Process Extension

Business Logic APIs

Business Process APIs

Data Aggregation APIs

1. List of items
2. Price and promos
3. Fulfillment options
4. Payment type

Store Tenant

Point of Sale

Store Checkout API Extension

Inventory Cache

Item-Price Cache

1. N/A
2. Store only promo
3. N/A
4. Chip and Pin

Checkout API

1. List of items
2. Price and promos
3. Fulfillment options
4. Payment type
Retail Platform Architecture

Process Extension

Digital Tenant

1. Retrieve prior cart
2. Digital coupon
3. Address or Pickup?
4. Paypal, Bitcoin

Tenants

- Target.com
- Digital Checkout API Extension

Platform

- Business Logic APIs
- Business Process APIs
- Data Aggregation APIs

- Inventory Cache
- Item-Price Cache

Checkout API

1. List of items
2. Price and promos
3. Fulfillment options
4. Payment type
Generic Platform Architecture

No Tenant to Tenant Calls
Retail Platform Architecture

All Transactions are Events
Retail Platform Architecture

Eventually Consistent

Tenants
- Tenant Services
- Forward Cache

Platform
- Business Logic APIs
- Business Process APIs
- Item-Price API

Merchandising Tenant
- Price Management
- Tenant Price Event Listener
- Item-Price Cache

Price API
- Price Update

Price Event Listener
- Price Event
- Item-Price Cache

Eventually Consistent
Platform Operating Principles

• No Tenant to Tenant calls
• All transactions are Events
• Events are (almost) completely open to all
• All caching layers are eventually consistent
Scalability Principles

• Protect Fundamental Data services
  • Platform Aggregations and Tenant caches
  • Throttle events to Fundamental Data layer
  • Asynchronous writes

• Serve majority of traffic from tenant layer
  • Distribute data to edge
  • All caching layers are eventually consistent
Handling Failure
Platform Micro-Failures

- Individual platform service failure
- Tenant to Platform network failure

Tenant decides how to handle failure
Retail Platform

Micro-failure - Read

503

Use last cached value
Indicate in UI price is old
Platform Macro-Failures

- Platform down
  - Network failures
  - DDOS
  - Exceeded capacity
  - Excessive latency

Tenant decides how to handle failure
1. Checkout down
2. Browse & Search OK
3. Store Locator OK
meet two of our newest brands, only at Target

A New Day

Goodfellow & Co
orders

August 23, 2017

picked up

August 20, 2017

delivered

delivered

delivered
Architecture Governance
Architecture Governance Considerations

- Every system has a context: Platform or Tenant
- Every system has a defined scope: Data, Process, Logic, Aggregation, UI
- Tenants are decoupled from other tenants & the platform
- Context decides amount of enterprise governance
Isolated impact
Open to new technologies
Build and deploy quickly
Tenants make many technology decisions

Retail Platform Architecture

Tenant Governance
Wide impact
Strong technology standards
Durable API contracts
Enterprise decision process

Retail Platform Architecture

Platform Governance
Architecture Considerations

• Adopt new technologies in Tenants
• Innovate and experiment in Tenants
• Learn and evaluate for graduation to Platform
Architecture Framework
Platform Architecture Framework

1. Architecture vision definition
2. Platform diagram – one page!
3. Top down agreement to build a platform
4. Communicate the strategy!
5. Define fundamental data
6. Define platform surface
7. Create Portfolio/Domain level diagrams
8. Create automated measurement of progress
#1 Architecture Vision

- Document it!
- Write a 10 page whitepaper

---

Target Technology 2019: Target is the Platform
Author: Joel Crabb

**Introduction**

**Objectives and Key Results**

We are a product driven organization, it would be remiss not to have OKRs for this paper.

**Objective:** Define a Service Oriented Architecture (SOA) path forward to a Target platform.

**Key Result:** 100% of new development is done to APIs.

**Key Result:** Average time between code commit to production deployment decreases by 20%.

**Key Result:** Ability to compose at least one view application from existing services.
#2 Platform Diagram – One Pager
#3 Top Down Agreement

- Reviewed with CIO and all VPs in IT
- Enterprise agreement on the platform direction
- Acknowledge it will require organizational restructuring

- Architecture team to evangelize and measure progress
#4 Communicate the Strategy

To reach 3000+ Engineers:

1. Have a 30 minute presentation on the platform
2. Take a lot of questions
3. Do it over and over and over
4. Teach the architects and the evangelists
5. Make a video
6. Continuous communication throughout the life of the platform
#5 Define Fundamental Data

- Identified 41 fundamental data topics
- Assigned ownership to technology teams
- Stand up the 10 most important APIs first
#6 Define Platform Surface

**Survey**
- Find existing APIs
- Define Needs

**Identify**
- Catalogue APIs
- Apply Metadata
  - Platform
  - Tenant

**Standardize**
- Event Metadata
- API Specification
- API Templates
#8 Measure Progress - Automatically

- Fundamental Data API count
- # of Tenants
- # of Platform Logic and Process APIs
- # of Events defined
- # of API calls
- # of Events generated
Platform Learnings
Learnings

• Fundamental Data systems need an event listener with business logic to decide what to Insert/Update/Delete
• Platform components need separate funding
• Once you establish a Platform model, everything becomes a platform
  • Infrastructure platform, pricing platform, guest data platform, identity platform, etc
Security Considerations

- Event security and provenance
- Event traceability
- Event data access authorization
- Event data encryption for PII
- API security and authorization
Issues to Work Through

• No Tenant to Tenant traffic
• South -> North traffic for large payloads
• Tenant granularity
• Tenants can become miniature monoliths
• Graduating tenant functions to the platform
• Aggregation proliferation at Tenant level
• Event definition
• Metadata definition
Use Conway’s Law

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization’s communication structure.

~ M. Conway

http://www.melconway.com/Home/Conways_Law.html

This is exactly what we want!
mobilePOS

Scan or key first item to begin.

mobilePOS

Luxury solid bath towel 12.00

Subtotal: $12.00
Discounts: $0.78
Tax: $0.78
Total: $12.78

next
Questions