7 YEARS OF DDD

or

Tackling Complexity in Large Scale Marketing Systems
YAY!!!
PART 1
5 BOUNDED CONTEXTS

PART 2
5 PRACTICAL ADVICES
THE FIRST BOUNDED CONTEXT
VER 1.0

- Media Buying
- Creatives Catalog
- Campaign Management

#OReillySACon

vladikk
Design
Tackling Complexity in the Heart of Software

Eric Evans

Foreword by Martin Fowler
Very nice!!!1
Aggregates everywhere!!!
Aggregates everywhere!!!
<table>
<thead>
<tr>
<th>Creative</th>
<th>Ad Type</th>
<th>Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency</td>
<td>Target Market</td>
<td>Ad Zone</td>
</tr>
<tr>
<td>Advertiser</td>
<td>Group</td>
<td>Contract</td>
</tr>
<tr>
<td>Publisher</td>
<td>Zone Type</td>
<td>Budget Unit</td>
</tr>
<tr>
<td>Website</td>
<td>Funnel</td>
<td>Audience</td>
</tr>
</tbody>
</table>
Imperfect architecture

“QA is for cowards”

BUT IT WORKED!
UBIQUITOUS LANGUAGE
Smooth communication

Strong grasp of the business domain

Working software

Aggressive time to market
Ubiquitous Language → Anemic Domain Model
OPTIMIZE

ALL THE THINGS
Lead qualification

Agent qualification

Agents’ commissions
THE CRM BOUNDED CONTEXT

02
Ubiquitous Language → Anemic Domain Model
<table>
<thead>
<tr>
<th>Creative</th>
<th>Ad Type</th>
<th>Advertiser</th>
<th>CRM Lead</th>
<th>Organization Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency</td>
<td>Target Market</td>
<td>Ad Zone</td>
<td>Group</td>
<td>Assignment</td>
</tr>
<tr>
<td>Marketing Lead</td>
<td>Group</td>
<td>Contract</td>
<td>Desk</td>
<td>Rank</td>
</tr>
<tr>
<td>Publisher</td>
<td>Zone Type</td>
<td>Budget Unit</td>
<td>Qualification</td>
<td>Message</td>
</tr>
<tr>
<td>Website</td>
<td>Funnel</td>
<td>Audience</td>
<td>Assessment</td>
<td>On-site Activity</td>
</tr>
<tr>
<td>Placement</td>
<td>Marketing Campaign</td>
<td>Visit</td>
<td>CRM Campaign</td>
<td>Brand</td>
</tr>
</tbody>
</table>
Software Developers

Domain Experts

…. Lead ....

…. Campaign...

#OREillySACon
Bounded Context

Figure 14.1 A navigation map for model integrity patterns

Bounded Context

Cells can exist because their membranes define what is in and out and determine what can be.

Multiple models coexist on big projects, and this works fine in many cases. Different models apply in different contexts. For example, you may have to integrate your new software with an external system over which your team has no control. A situation like this is probably clear to everyone as a distinct context where the model doesn’t apply, but other situations can be more vague and evolving. In the functionality for the same new system, were they working on the same models? Their intention was to share at least part of what they did, but there was no demarcation to tell them what they did or did not share. And they had no process in place to hold a shared model together or quickly detect divergences. They realized they had diverged only after their system’s behavior suddenly became unpredictable.

Even a single team can end up with multiple models. Communication can lapse, leading to subtly conflicting interpretations of the model. Older code often reflects an earlier conception of the model that is subtly different from the current model.

Everyone is aware that the data format of another system is different and calls for a data conversion, but this is only the mechanical dimension of the problem. More fundamental is the difference in the
Achievement unlocked
Read the blue book
BOUNDDED CONTEXTS
PROTECT THE LANGUAGE
AGGREGATES PROTECT
CONSISTENCY OF DATA
<table>
<thead>
<tr>
<th>Creative</th>
<th>Ad Type</th>
<th>Advertiser</th>
<th>CRM Lead</th>
<th>Organization Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency</td>
<td>Target Market</td>
<td>Ad Zone</td>
<td>Group</td>
<td>Assignment</td>
</tr>
<tr>
<td>Marketing Lead</td>
<td>Group</td>
<td>Contract</td>
<td>Desk</td>
<td>Rank</td>
</tr>
<tr>
<td>Publisher</td>
<td>Zone Type</td>
<td>Budget Unit</td>
<td>Qualification</td>
<td>Message</td>
</tr>
<tr>
<td>Website</td>
<td>Funnel</td>
<td>Audience</td>
<td>Assessment</td>
<td>On-site Activity</td>
</tr>
<tr>
<td>Placement</td>
<td>Marketing Campaign</td>
<td>Visit</td>
<td>CRM Campaign</td>
<td>Brand</td>
</tr>
</tbody>
</table>
### Marketing

<table>
<thead>
<tr>
<th>Creative</th>
<th>Ad Type</th>
<th>Advertiser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency</td>
<td>Target Market</td>
<td>Ad Zone</td>
</tr>
<tr>
<td>Lead</td>
<td>Group</td>
<td>Contract</td>
</tr>
<tr>
<td>Publisher</td>
<td>Zone Type</td>
<td>Budget Unit</td>
</tr>
<tr>
<td>Website</td>
<td>Funnel</td>
<td>Audience</td>
</tr>
<tr>
<td>Placement</td>
<td>Campaign</td>
<td>Visit</td>
</tr>
</tbody>
</table>

### CRM

<table>
<thead>
<tr>
<th>Lead</th>
<th>Organization Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Assignment</td>
</tr>
<tr>
<td>Desk</td>
<td>Rank</td>
</tr>
<tr>
<td>Qualification</td>
<td>Message</td>
</tr>
<tr>
<td>Assessment</td>
<td>On-site Activity</td>
</tr>
<tr>
<td>Campaign</td>
<td>Brand</td>
</tr>
</tbody>
</table>
Aggregates will:

• Protect transactional boundaries

• Encompass business logic and invariants
Aggregates will:

• Protect transactional boundaries

• Encompass business logic and invariants
Stored procedures???
<table>
<thead>
<tr>
<th>Lead</th>
<th>Organization Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Assignment</td>
</tr>
<tr>
<td>Desk</td>
<td>Rank</td>
</tr>
<tr>
<td>Qualification</td>
<td>Message</td>
</tr>
<tr>
<td>Assessment</td>
<td>On-site Activity</td>
</tr>
<tr>
<td>Campaign</td>
<td>Brand</td>
</tr>
</tbody>
</table>
Achievement unlocked
Pwned by the Conway's Law
Inconsistent models

No shared understanding

Duplication of knowledge

Went out of sync quickly

NIGHTMARE
Wasn’t delivered on time

Production issues

Data corruption

Thrown away and reimplemented
Ubiquitous Language → Bounded Contexts → Domain Model

Protect with → Implement as

...Dude, where are Sub-Domains?
Not all of a large system will be well designed

""

Eric Evans
THE CRUNCHERS
BOUNDDED CONTEXT

03
Customer Events → Event Crunchers → Marketing
Customer Events → Event Crunchers → CRM
Customer Events → Event Crunchers → Analysis
Competitive advantage? - No

Off-the-shelf solution? - No

=> Supporting sub-domain
Layered Architecture

Transaction Script

Worked

.... for a while
THE BONUSES
BOUNDDED CONTEXT

04
Sales → Commissions → Reports
Competitive advantage? - No

Off-the-shelf solution? - No

=> Supporting sub-domain
Let’s try different percentages

No, what if the percentage is a function of number of sales?

No, no, the percentage will be a function both of number of sales and sale amount

What if the percentage could be a function of a price?

And another upgrade if there are more than Y sales per week!

But we will upgrade the percentage if there are more than X sales per month!
Event Crunchers

- Infrastructure
- Transaction Script
- Service / Application Layer
- Presentation

Bonuses

- Infrastructure
- Active Record
- Service / Application Layer
- Presentation

Ubiquitous Language
THE MARKETING HUB
BOUNDDED CONTEXT

05
Competitive advantage? - Yes

=> Core Domain
Event Sourced Domain Model

CQRS

Microservices
Event Sourced Domain Model

CQRS

Microservices
TECHNICAL COMPLEXITY > BUSINESS COMPLEXITY
WHAT WE HAVE LEARNED
UBIQUITOUS LANGUAGE
UBIQUITOUS LANGUAGE
THE CORE DOMAIN OF DOMAIN-DRIVEN DESIGN
Marketing
✓ Ubiquitous Language
✓ Business goals achieved

CRM
✓ Ubiquitous Language
✓ Production issues
✓ Long and painful refactoring

Event Crunchers
✓ Ubiquitous Language
✓ Big ball of mud

Bonuses
✓ Ubiquitous Language
✓ Refactored in time
Invest in the Ubiquitous Language early on
Marketing

✓ Ubiquitous Language
✓ Business goals achieved

Event Crunchers

→ Ubiquitous Language
→ Big ball of mud

CRM

→ Ubiquitous Language
→ Production issues
→ Long and painful refactoring

Bonuses

✓ Ubiquitous Language
✓ Refactored in time
Core

Supporting

Generic
Core Domain → Domain Model / Event Sourcing

Supporting Domain → Active Record / Transaction Script

Generic Domain → Adopt / Buy
COMPANIES CHANGE, EVOLVE, REINVENT THEMSELVES

DOMAINS' TYPES CHANGE ACCORDINGLY
SUPPORTING ➤ CORE
• Event Crunchers
• Bonuses

SUPPORTING ➤ GENERIC
• Creative Catalog

CORE ➤ GENERIC
• Lead Evaluation System

CORE ➤ SUPPORTING
• Marketing Hub

GENERIC ➤ CORE
• AWS

#OREillySACon
LESS WASTE

IMPLEMENTATION DESIGN ➤ DOMAIN TYPE

Less waste

Dialog with the business
BUSINESS COMPLEXITY ≠ DOMAIN TYPE?

- Questionable competitive edge?
- Accidental “business” complexity?
- Unexpected competitive edge?
IMPLEMENTATION DESIGN ➤ DOMAIN TYPE

- Domain Model / Event Sourcing ➔ Core Domain
- Active Record / Transaction Script ➔ Supporting Domain
- Adopt / Buy ➔ Generic Domain
IMPLEMENTATION STRATEGIES
How to Model the Business Logic?
How to Model the Business Logic?

- Transaction Script (PoEAA)
- Active Record (PoEAA)
- Domain Model (PoEAA + DDD)
- Event Sourced Domain Model
MONEY? DEEP ANALYTICS? AUDIT LOG?
Event Sourced Domain Model

COMPLEX BUSINESS LOGIC?
Domain Model

COMPLEX DATA STRUCTURES?
Active Record

SIMPLE LOGIC, SIMPLE DATA STRUCTURES?
Transaction Script
MAPPING ARCHITECTURAL PATTERNS

Event Sourced Domain Model ➤ CQRS

Domain Model ➤ Hexagonal Architecture

Active Record ➤ Layered Architecture

Transaction Script ➤ “Keep it simple” Architecture
Transaction Script

Active Record

Domain Model

Event Sourced Domain Model
PAIN?

BUSINESS CHANGED?

DOMAIN TYPE CHANGED?

REVISE IMPLEMENTATION STRATEGY?
Transaction Script

Active Record

Domain Model

Event Sourced Domain Model
BOUNDDED CONTEXTS
DOMAIN-BASED BOUNDARIES

Event Crunchers

Bonuses
AGGREGATE-BASED BOUNDARIES
SUICIDAL BOUNDARIES

Lead

Lead
Good Fences: The Importance of Setting Boundaries for Peaceful Coexistence

Alex Rutherford, Dion Harmon, Justin Werfel, Alexander S. Gard-Murray, Shlomiya Bar-Yam, Andreas Gros, Ramon Xulvi-Brunet, Yaneer Bar-Yam*

New England Complex Systems Institute, Cambridge, Massachusetts, United States of America

Abstract

We consider the conditions of peace and violence among ethnic groups, testing a theory designed to predict the locations of violence and interventions that can promote peace. Characterizing the model's success in predicting peace requires examples where peace prevails despite diversity. Switzerland is recognized as a country of peace, stability and prosperity. This is surprising because of its linguistic and religious diversity that in other parts of the world lead to conflict and violence. Here we analyze how peaceful stability is maintained. Our analysis shows that peace does not depend on integrated coexistence, but rather on well defined topographical and political boundaries separating groups, allowing for partial autonomy within a single country. In Switzerland, mountains and lakes are an important part of the boundaries between sharply defined linguistic areas. Political canton and circle (sub-canton) boundaries often separate religious groups. Where such boundaries do not appear to be sufficient, we find that specific aspects of the population distribution guarantee either
Finding service boundaries is really damn hard… There is no flowchart!

“"

Udi Dahan
THERE ARE GOING TO BE MISTAKES
ACCEPT IT AND DON’T MAKE FATAL ONES
START WITH BIGGER BOUNDARIES
DECOMPOSE LATER, AS YOU GAIN KNOWLEDGE
THE LESS YOU KNOW ABOUT THE DOMAIN
THE WIDER THE INITIAL BOUNDARIES
1. Ubiquitous Language is not optional

2. Domain Types change. Embrace these changes to achieve resilient design

3. Learn the ins and outs of the four patterns of modeling business logic

4. Always start with bigger boundaries, but decompose further as you gain domain knowledge
Ubiquitous Language

Protect by decomposing to

Bounded Contexts

Design Implementation

Domain Model

Core

AR / TS

Supporting

Adopt

Generic

Confirm w/ Business

Supporting
Protect by decomposing to Bounded Contexts

Design Implementation

Domain Model

Confirm w/ Business

Core

AR / TS

Supporting

Generic

Ubiquitous Language

Adopt
Ubiquitous Language

Protect by decomposing to

Bounded Contexts

**Design Implementation**

- Domain Model
- AR / TS
- Adopt

**Confirm w/ Business**

- Core
- Supporting
- Generic

#OReillySACon
Aggregates everywhere!!!
Ubiquitous Language Everywhere!!!

Protect by decomposing to Bounded Contexts to Design Implementation to Confirm w/ Business

Ubiquitous Language

Bounded Contexts

Domain Model

Core

AR / TS

Supporting

Adopt

Generic

Ubiquitous Language Everywhere!!!
THANK YOU!

@vladikk

vladikk.com


http://careers.doit-intl.com

https://bit.ly/31nSD9c
Rate today’s session

Session page on oreillysacon.com/ny

O’Reilly Events App