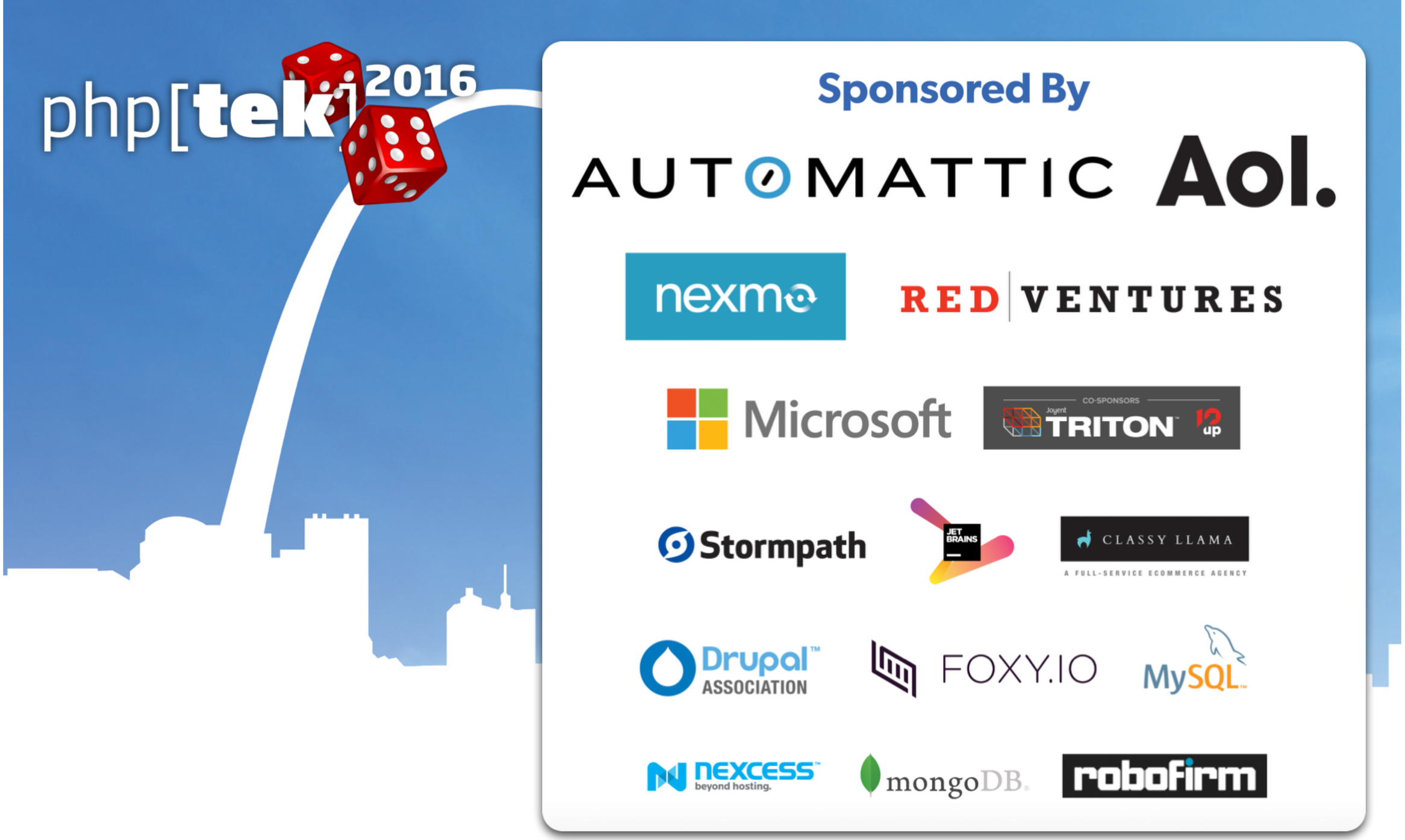
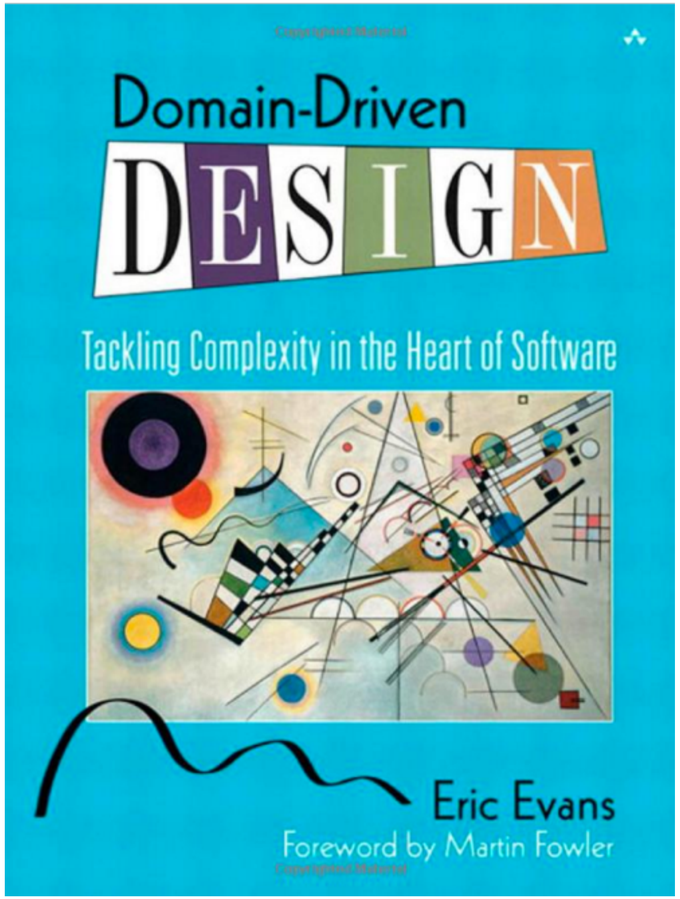
Domain-driven Design Deconstructed

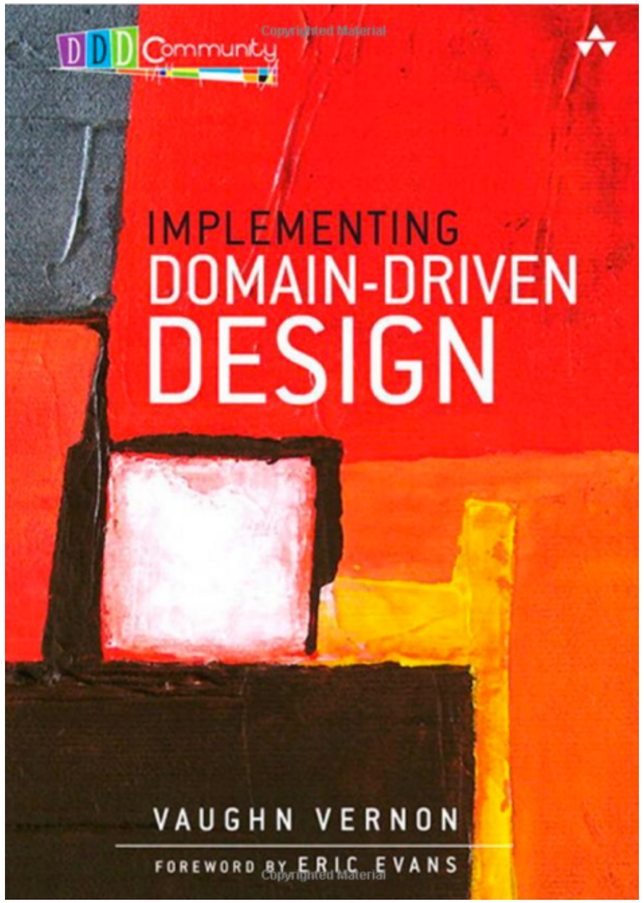
Andrew Cassell

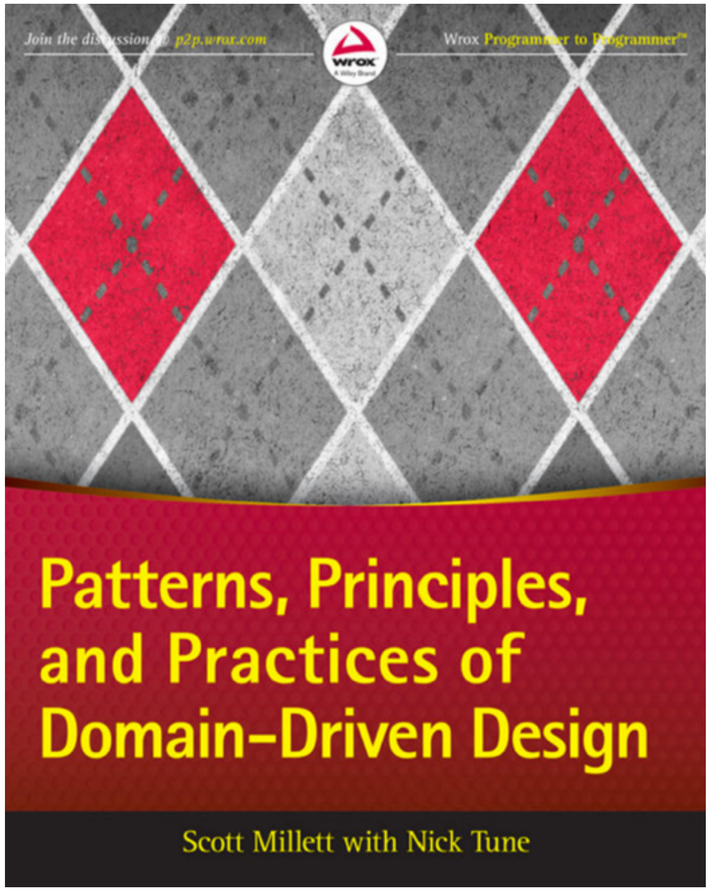
php
www.phparch.com
Fill your glass with
php[architect]

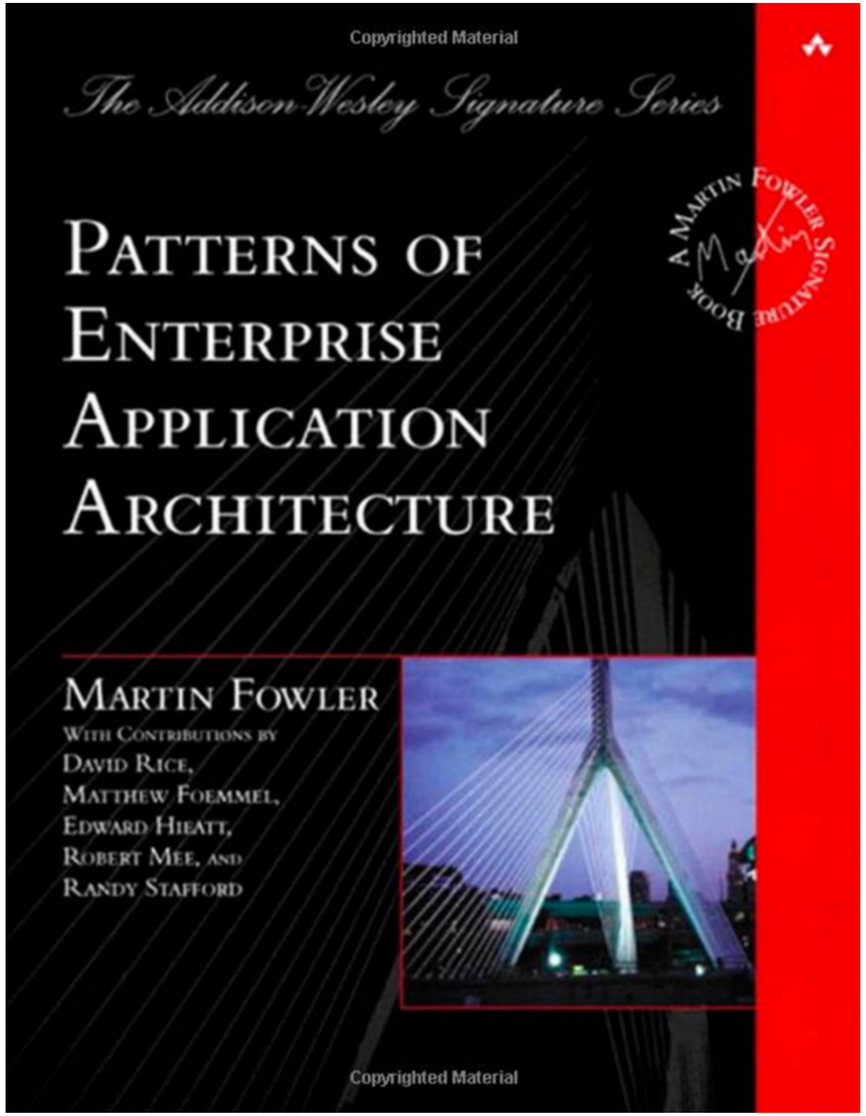

php[tek] 2016
Sponsored By
AUTOMATTIC
Aol.
nexmo
RED VENTURES
Microsoft
CO-SPONSORS
Joyent TRITON
Stormpath
JET BRAINS
CLASSY LLAMA
A FULL-SERVICE ECOMMERCE AGENCY
Drupal ASSOCIATION
FOXY.IO
MySQL
NEXCESS beyond hosting.
mongoDB
robofirm

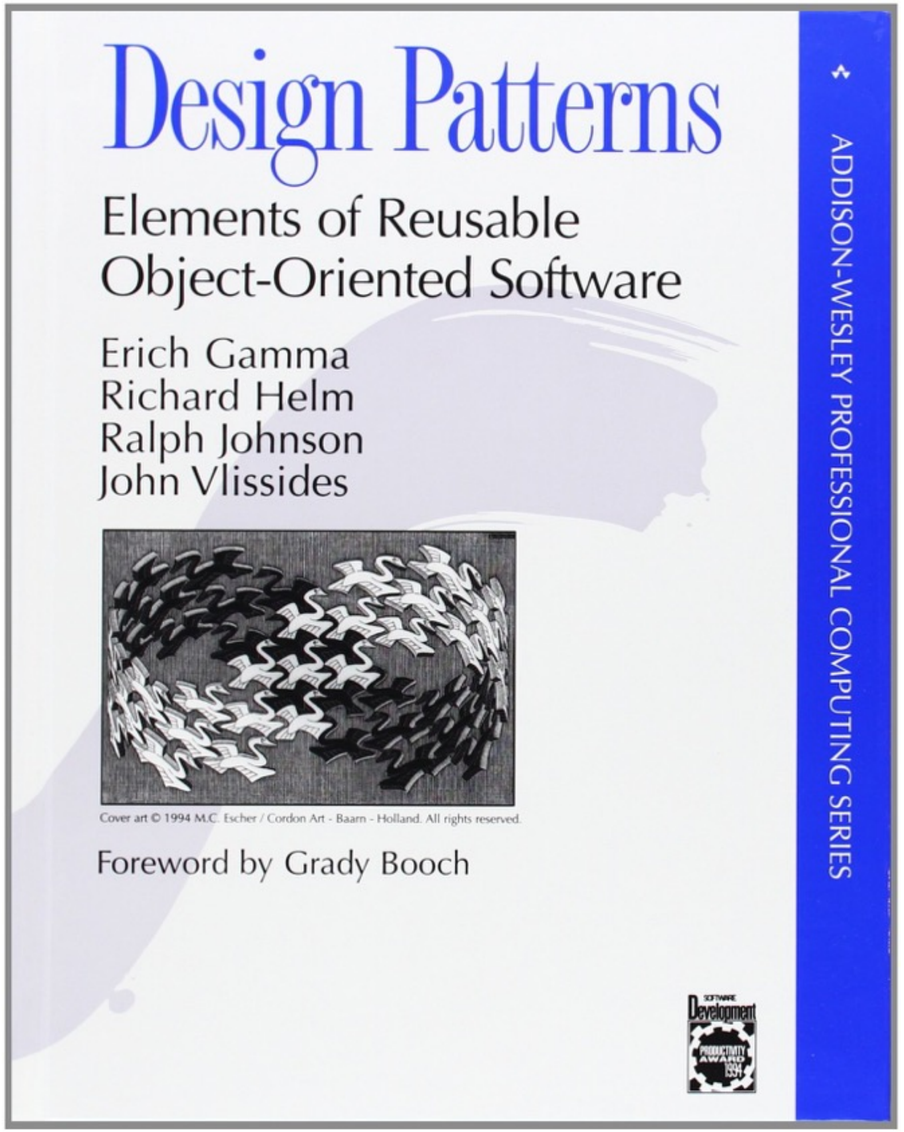
Domain-driven Design

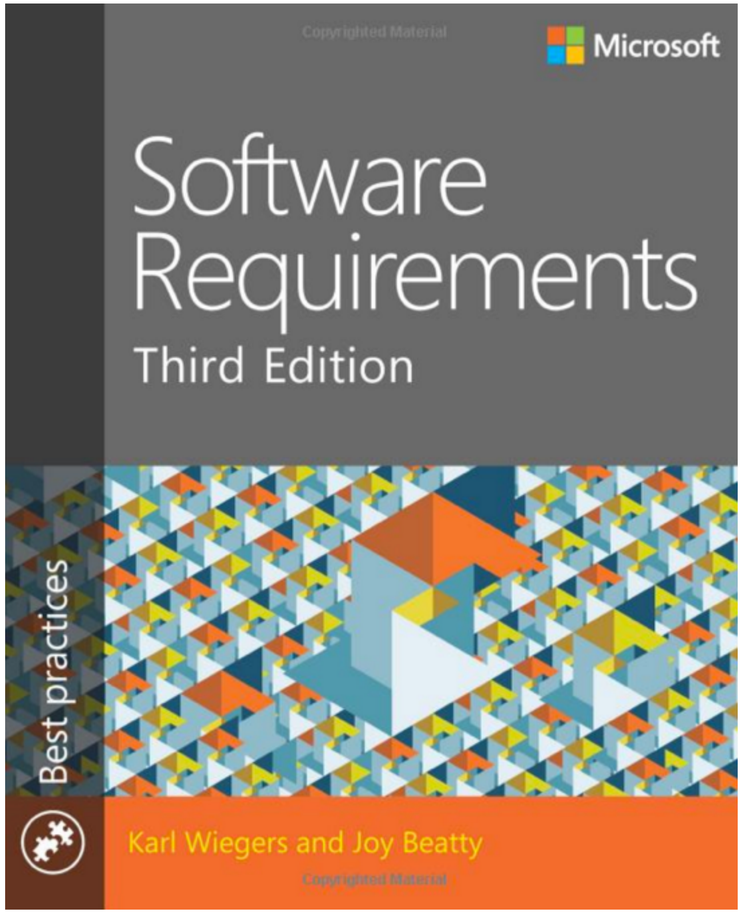

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

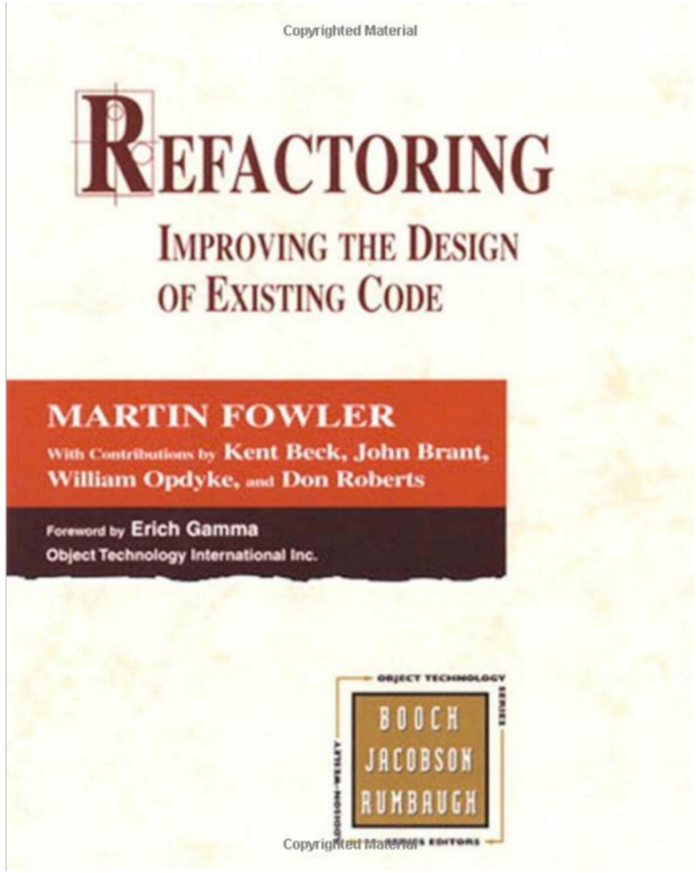

DDD Community
IMPLEMENTING
DOMAIN-DRIVEN
DESIGN
VAUGHN VERNON
FOREWORD BY ERIC EVANS


Wrox
Patterns, Principles,
and Practices of
Domain-Driven Design
Scott Millett with Nick Tune


Copyrighted Material
The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Patterns of Enterprise Application Architecture
Martin Fowler
With Contributions by
David Rice,
Matthew Foemmel,
Edward Hieatt,
Robert Mee, and
Randy Stafford
Copyrighted Material

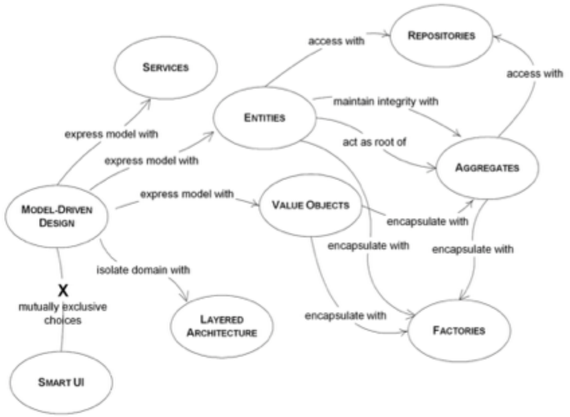

Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


Microsoft
Software Requirements
Third Edition
Best practices
Karl Wiegers and Joy Beatty


Copyrighted Material
REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International Inc.
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH

A Summary of Eric Evans' ***Domain-Driven Design***

Domain-Driven Design *Quickly*



by Abel Avram & Floyd Marinescu

edited by: Dan Bergh Johnsson, Vladimir Gitlevich

InfoQ *Enterprise Software Development Series*

DOMAIN-DRIVEN DESIGN

IN PHP

CARLOS BUENOSVINOS
CHRISTIAN SORONELLAS
KEYVAN AKBARY

Microsoft

Exploring CQRS and Event Sourcing

A journey into high scalability, availability and maintainability with Windows Azure

Dominic Betts
Julian Dominguez
Grigori Melnik
Fernando Simonazzi
Mani Subramanian

Foreword by Greg Young

July 2012

patterns & practices

Mathias Verraes

mathiasverraes

Value Object Comm.V

Belgium

http://verraes.net

Beau Simensen

simensen

dflydev

Madison, WI

beau@dflydev.com

https://beau.io

Domain-driven Design

DDD

Cal Evans
@CalEvans

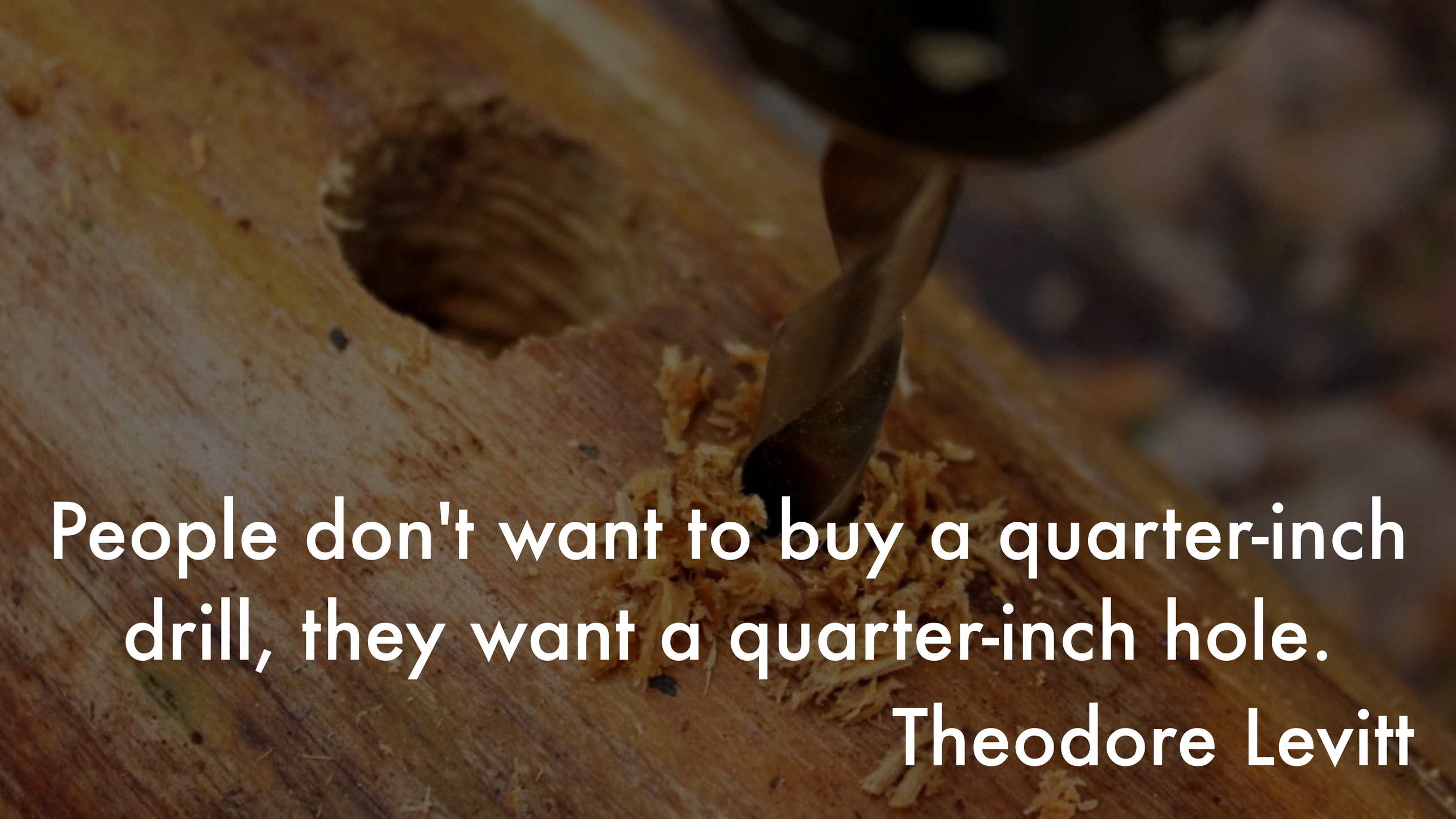
OH “What is #DDD?”

Drink, Drive, Develop.

#phpbnl16

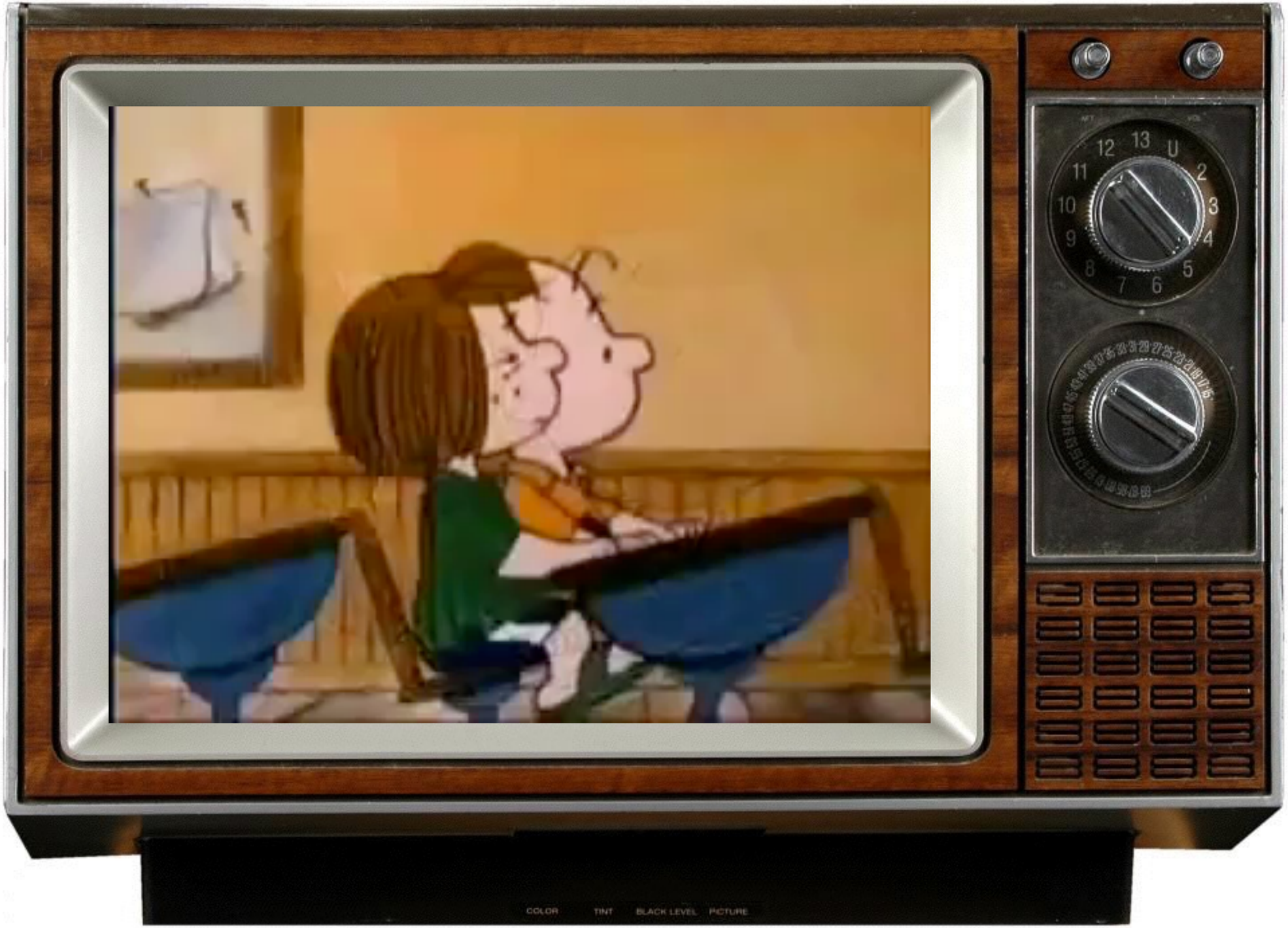
11:00 AM - 28 Jan 2016

SNAP A LOOP™
www.SnapALoop.com
1-800-820-0003



People don't want to buy a quarter-inch drill, they want a quarter-inch hole.

Theodore Levitt


COLOR
TINT
BLACK LEVEL
PICTURE

Domain-driven design

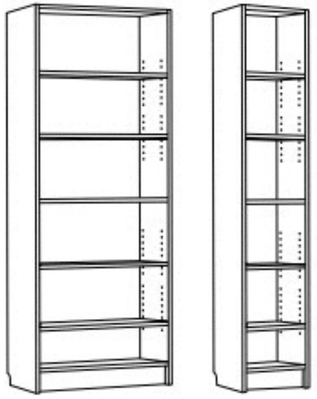
From Wikipedia, the free encyclopedia

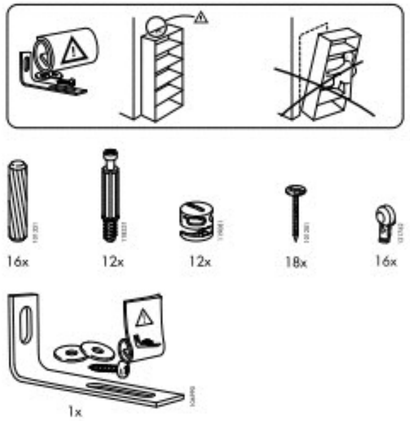
Domain-driven design (**DDD**) is an approach to software development for complex needs by connecting the implementation to an evolving model.[1] The premise of domain-driven design is the following:

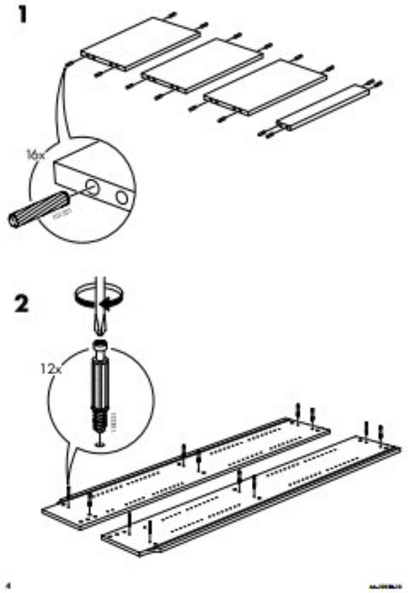
- placing the project's primary focus on the core domain and domain logic;
- basing complex designs on a model of the domain;
- initiating a creative collaboration between technical and domain experts to iteratively refine a conceptual model that addresses particular domain problems.

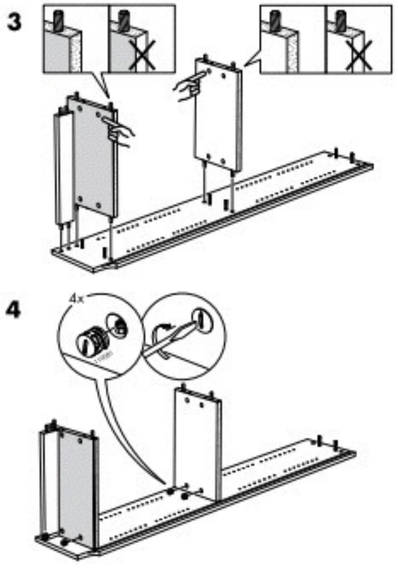
The term was coined by Eric Evans in his book of the same title.[2]

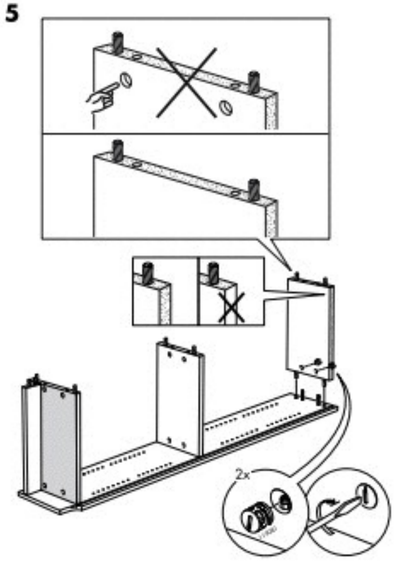
BILLY

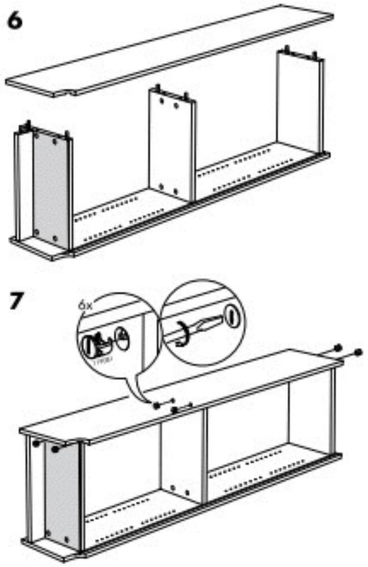


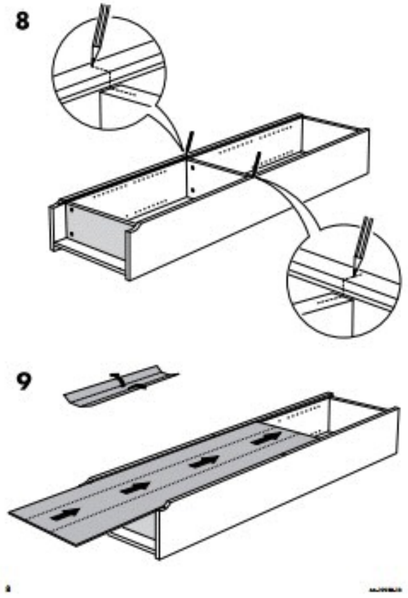






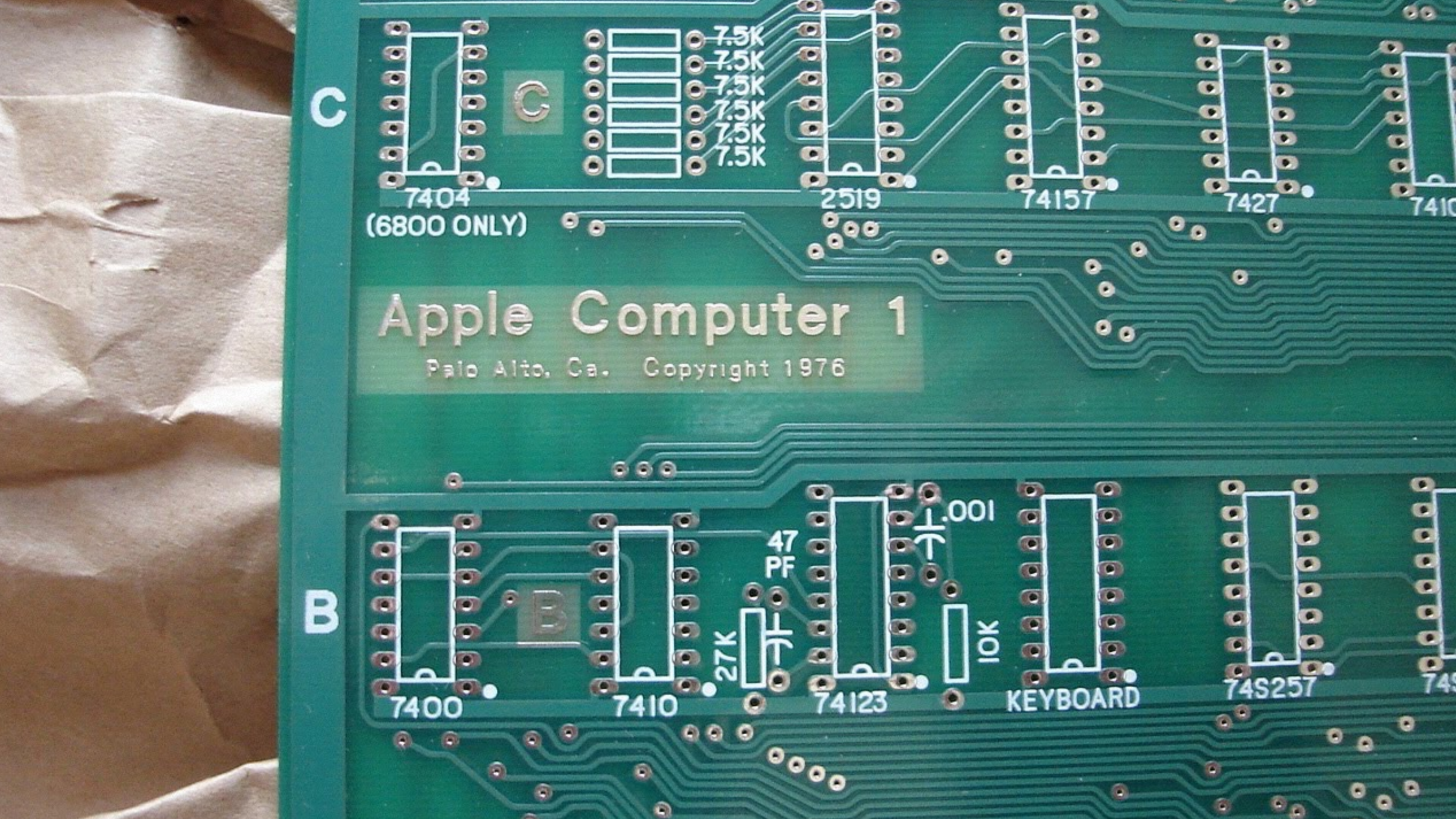






IKEA®
Design and Quality
IKEA of Sweden

#DDD4ME


C
7404
(6800 ONLY)
C
7.5K
7.5K
7.5K
7.5K
7.5K
7.5K
2519
74157
7427
Apple Computer 1
Palo Alto, Ca. Copyright 1976
B
7400
B
7410
27K
47
PF
74123
.001
10K
KEYBOARD
74S257

CRAFTSMAN®

DEPTH
GAUGE
90 ft

Common Sense Software Development

Domain

Real Spaces
DIPLOMACY BY DESIGN
AMERICAN ART

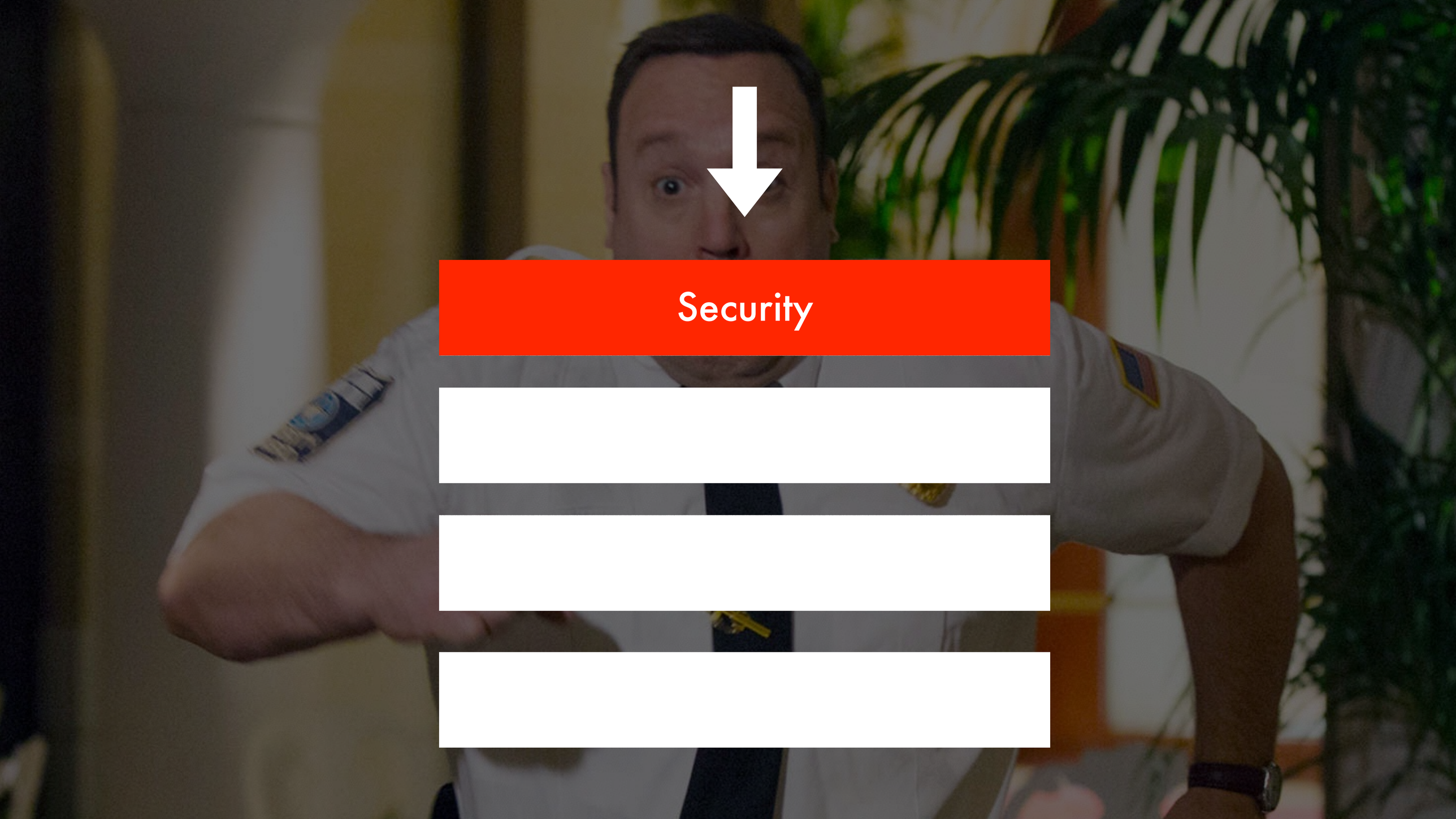
http://www.sei.cmu.edu/reports/95tr021.pdf

Table 1: Comparison between several broad Quality Taxonomies

McCall et. al.	RADC	Wiegers
Correctness	Correctness	
Reliability	Reliability	Reliability
Usability	Usability	Usability
Integrity	Integrity	Integrity
Efficiency	Efficiency	Efficiency
Portability	Portability	*Portability*
Reusability	Reusability	*Reusability*
Interoperability	Interoperability	Interoperability
Maintainability	Maintainability	*Maintainability*
Flexibility	Flexibility	Flexibility
Testability	Verifiability	*Testability*
	Survivability	Robustness
	Expandability	
		Installability
		Safety
		Availability

	DDD	BALL OF MUD/CRUD
Correctness		
Testability		
Usability		
Maintainability		
Modifiability		

Performance



Security

DDD Common Misconceptions

INITECH

TopGear.com

MOVE
FAST AND
BREAK
THINGS

MOVE FAST WITH STABLE INFRA

Dunland
Isengard
North-South Road
Fords of Isen
WESTFOLD
Emnet
Helm's Deep
Snowbourn
Middle Earth

WENGER
OFFICIAL
WORLD SCOUT KNIFE

This repository Search Pull requests Issues Gist

EnterpriseQualityCoding / **FizzBuzzEnterpriseEdition**

Watch 138 Star 5,306 Fork 288

Code Issues 128 Pull requests 24 Wiki Pulse Graphs

FizzBuzz Enterprise Edition is a no-nonsense implementation of FizzBuzz made by serious businessmen for serious business purposes.
http://www.fizzbuzz.enterprises

133 commits **2** branches **0** releases **24** contributors

Branch: **master** New pull request Create new file Upload files Find file Clone or download

Mikkeren Reverting the revert commit, since it clearly did not revert as inten… ··· Latest commit 6f865cc on Oct 26, 2015

resources/assets/configuration/spring/de…	Installed spring and restructured for dependency injection.	a year ago
src	Reverting the revert commit, since it clearly did not revert as inten…	7 months ago
.gitattributes	Started the project. Time to learn!	4 years ago
.gitignore	Installed spring and restructured for dependency injection.	a year ago
.travis.yml	Reverting the revert commit, since it clearly did not revert as inten…	7 months ago
CONTRIBUTING.md	Reverting the revert commit, since it clearly did not revert as inten…	7 months ago
README.md	Reverting the revert commit, since it clearly did not revert as inten…	7 months ago

Table 1.1 The DDD Scorecard

Does Your Project Score a Total of 7 Points or Higher?

If Your Project . . .	Points	Supporting Thoughts	Your Score
If your application is completely data-centric and truly qualifies for a pure CRUD solution, where every operation is basically a simple database query to Create, Read, Update, or Delete, you don't need DDD. Your team just needs to put a pretty face on a database table editor. In other words, if you can trust your users to insert data directly into a table, update it, and sometimes delete it, you wouldn't even need a user interface. That's not realistic, but it's conceptually relevant. If you could even use a simple database development tool to create a solution, don't waste your company's time and money on DDD.	0	This seems like a no-brainer, but it's not usually that easy to determine simple versus complex. It's not as if every application that isn't pure CRUD deserves the time and effort of using DDD. So maybe we could come up with other metrics to help us draw a line between what is complex and what is not . . .	
If your system requires just 30 or fewer business operations, it's probably pretty simple. This would mean that your application would have no more than 30 total user stories or use case flows, with each of those flows having only minimal business logic. If you could quickly and easily develop such an application using Ruby on Rails or Groovy and Grails and not feel the pain of lacking power and control over complexity and change, your system probably doesn't need to use DDD.	1	To be clear, I am talking about 25 to 30 single business methods, not 25 to 30 whole service interfaces, each with multiple methods. The latter might be complex.	
So let's say that somewhere in the range of 30 to 40 user stories or use case flows could be creeping toward complexity. Your system might be getting into DDD territory.	2	*Caveat emptor*: Very often complexity is not recognized soon enough. *We software developers are really, really good at underestimating complexity and*	

CRUD is okay-ish

Excessive parameters in a DDD Use Case self.PHP

3

Submitted 11 months ago by EspadaV8

I'm working on a DDD project (in Laravel) and trying to write a use case (Command) but it looks like it's going to need a *lot* of parameters (around 19 at the moment). These are just the fields that are presented to the user in the form.

Is it okay to just pass these in? I thought about setting them in an array and just passing that in, but that removes any kind of type hinting or suggestion of what parameters are actually required, and is basically just hiding the parameters in the array.

Some of them could probably be combined into multiple DTOs, but again, does that really provide any benefit?

Of course, the values need to get from the form to the handler, I'm just not sure what the 'best' way to do this would be.

Thoughts, suggestions and questions are all very welcome :-)

Thanks

28 comments share save hide give gold report

all 28 comments

sorted by: best▼

[–] **utotwel** - Laravel Creator **5 points** 11 months ago

I'll be honest, I don't understand what benefit you think you are gaining here. Just pass a plain PHP array of the data you need and move on with your day. I know smart people will tell you that is not the "right way" but they are obsessing over pointless, trivial crap.

permalink embed save give gold

verraes.net/2013/04/crud-is-an-anti-pattern/

Mathias Verraes Services Workshops Agenda Blog Talks Contact

CRUD is an antipattern

CRUD doesn't express behaviour. Avoid setters, and use expressive, encapsulated operations instead.

Published on 19 April 2013 by @mathiasverraes

CRUD is, as Greg Young calls it, our industry's Grand Failure. We have deluded our users, and ourselves, that applications are nothing more than a thin layer around our database. We present the user with grids and edit forms, and ask them to fill in fields, that we map more or less directly to our relational tables. We use terms like "data model".

Real processes and businesses are not CRUD though. Real people don't say "I've set the paid amount of the order to '120', the paid currency to 'EUR', and status to 'paid'". They say "I've paid €120 for the order".

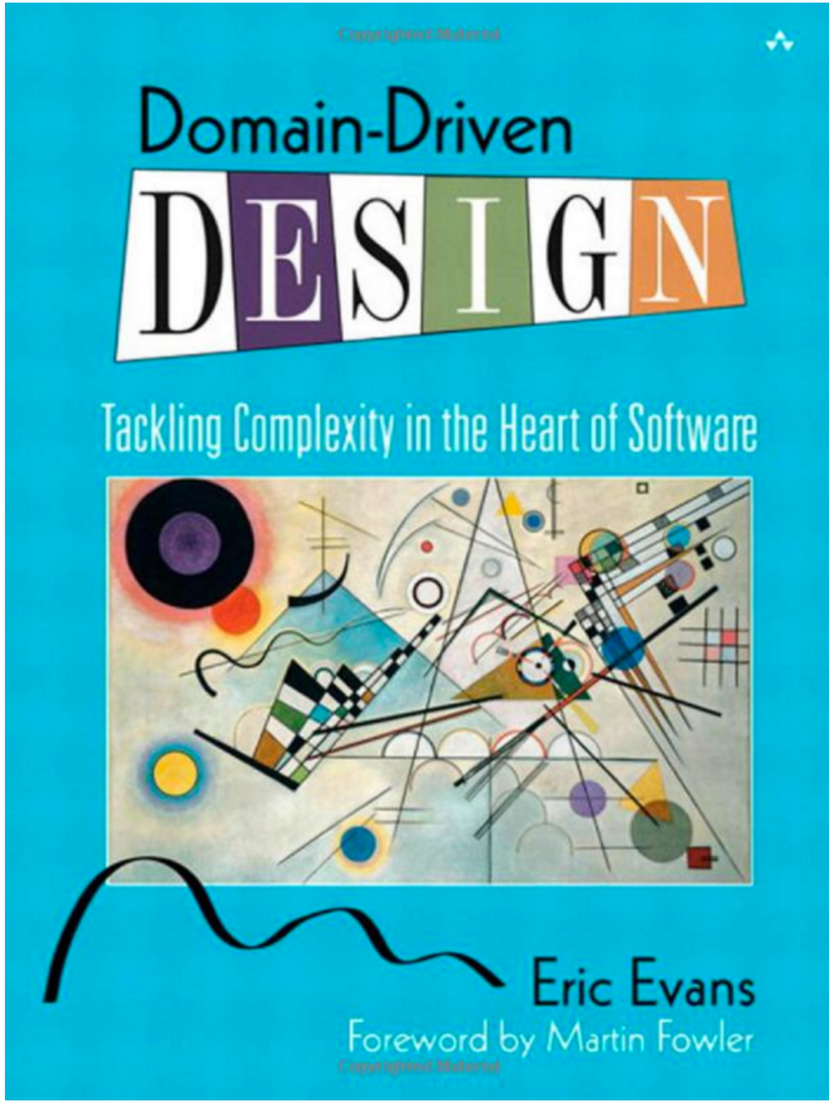
A good domain model is not a data model. A domain model describes behaviour, and the data is an artefact of

Andrzej Krzywda
@andrzejkrzywda

You know what's the biggest tragedy in software engineering?

The customers gave up and learnt to speak CRUD to developers.

7:33 AM - 3 Oct 2015


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Ubiquitous Language

For first you write a sentence,
And then you chop it small;
Then mix the bits, and sort them out
Just as they chance to fall:
The order of the phrases makes
No difference at all.
—*Lewis Carroll, "Poetic Fit, Non Nisi Scitur"*

To create a supple, knowledge-rich design, can for[illegible] ve[illegible]tile, sha[illegible] team language, and a live[illegible] experimentation [illegible] th[illegible] dom happens on software projects.

* * *

Domain experts have limited understanding of the technical jargon of software development, but they use the jargon of their field—probably in various flavors. Developers [illegible] understand and discuss the [illegible]stem [illegible] des[illegible]pti[illegible], fu[illegible]tio[illegible] terms devoid of the meaning carrie[illegible] [illegible] la[illegible]gua[illegible] Or[illegible] ers may create abstractions that support their design but a[illegible] derstood by the domain experts. Developers working on di[illegible] parts of the problem work out their own design concepts and ways of describing the domain.

Across this linguistic divide, the domain experts vaguely describe what they want. Developers, struggling to understand a domain new to them, vaguely understand. A few members of the team manage to become bilingual, but they become bottlenecks of information flow and their translations are inexact.

On a project without a common language, developers have to translate [illegible] between de-

A project faces serious problems when its language is fractured. Domain experts use their jargon while technical team members have their own language tuned for discussing the domain in terms of design.

The terminology of day-to-day discussions is disconnected from the terminology embedded in the code (ultimately the most important product of a software project). And even the same person uses differe[illegible] language in speech and in writing, so that the most incisive expressions [illegible] the domain often emerge in a transient form that is n[illegible]er [illegible]tu[illegible] in[illegible] co[illegible]r [illegible]en in [illegible]rit[illegible]

Tr[illegible]slat[illegible]n [illegible]unts [illegible]mu[illegible]nic[illegible]on [illegible] makes knowledge [illegible]unching a[illegible]mic.

Yet none of these dialects can be a common language because none serves all needs.

The overhead cost of all the translation, plus the risk of misunderstanding, is just too high. A project needs a common language that is more robust than the lowest common denominator. With a [illegible]ons[illegible]us [illegible] the [illegible] model can provide the back[illegible]one [illegible]r tha[illegible]on [illegible]on la[illegible]uag[illegible] [illegible]onnecting team communica[illegible]n t[illegible] s[illegible]war[illegible]mpl[illegible]ent[illegible]n. T[illegible]st language can be ubiquitous in the team's work.

The vocabulary o[illegible] [illegible]BIQUITOUS LANGUAGE includes the names of classes and prominent operations. The LANGUAGE includes terms to discuss rules that have been made explicit in the model. It is supplemented with terms from high-level organizing principles imposed on the model (such as CONTEXT MAPS and large-scale structures, which will be discussed in Chapters 14 and 16). Finally, this language is enriched with the names of patterns the team commonly applies to the domain model.

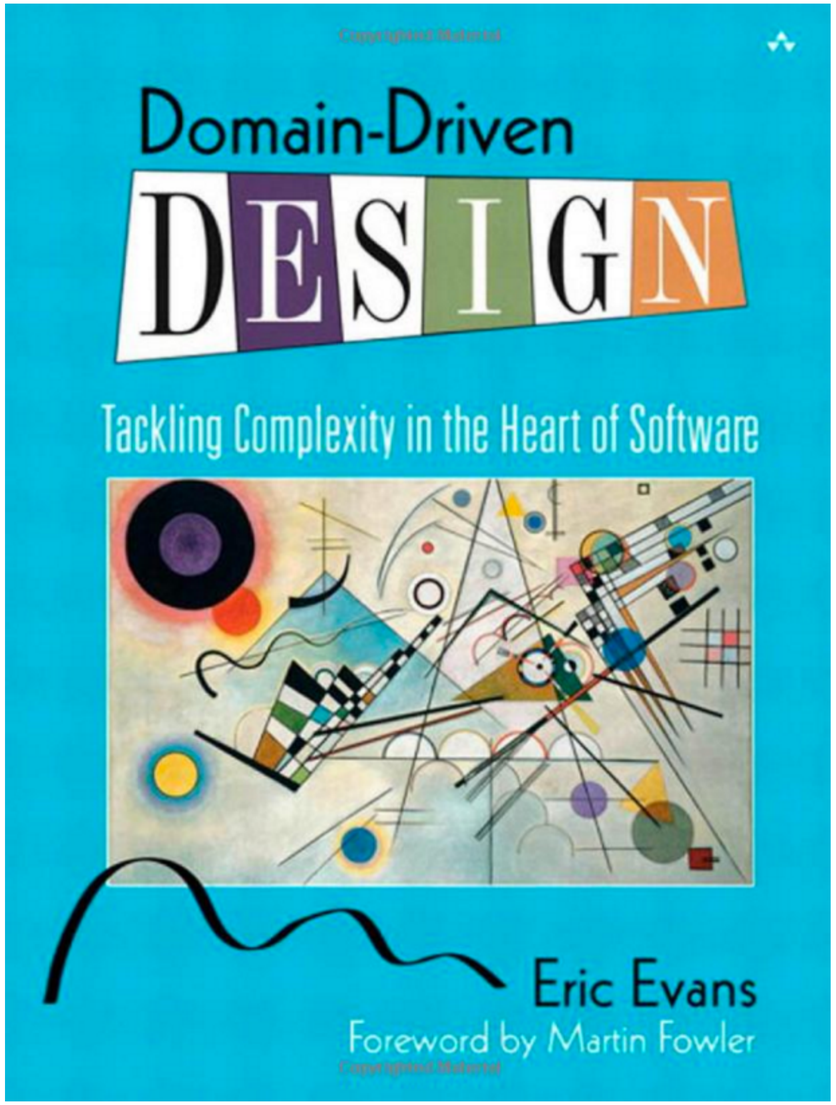
The model relationships become the combinatory rules all languages have. The meanings of words and phrases echo the semantics of the model.

Ubiquitous Language

Developers

Domain Experts

Octan



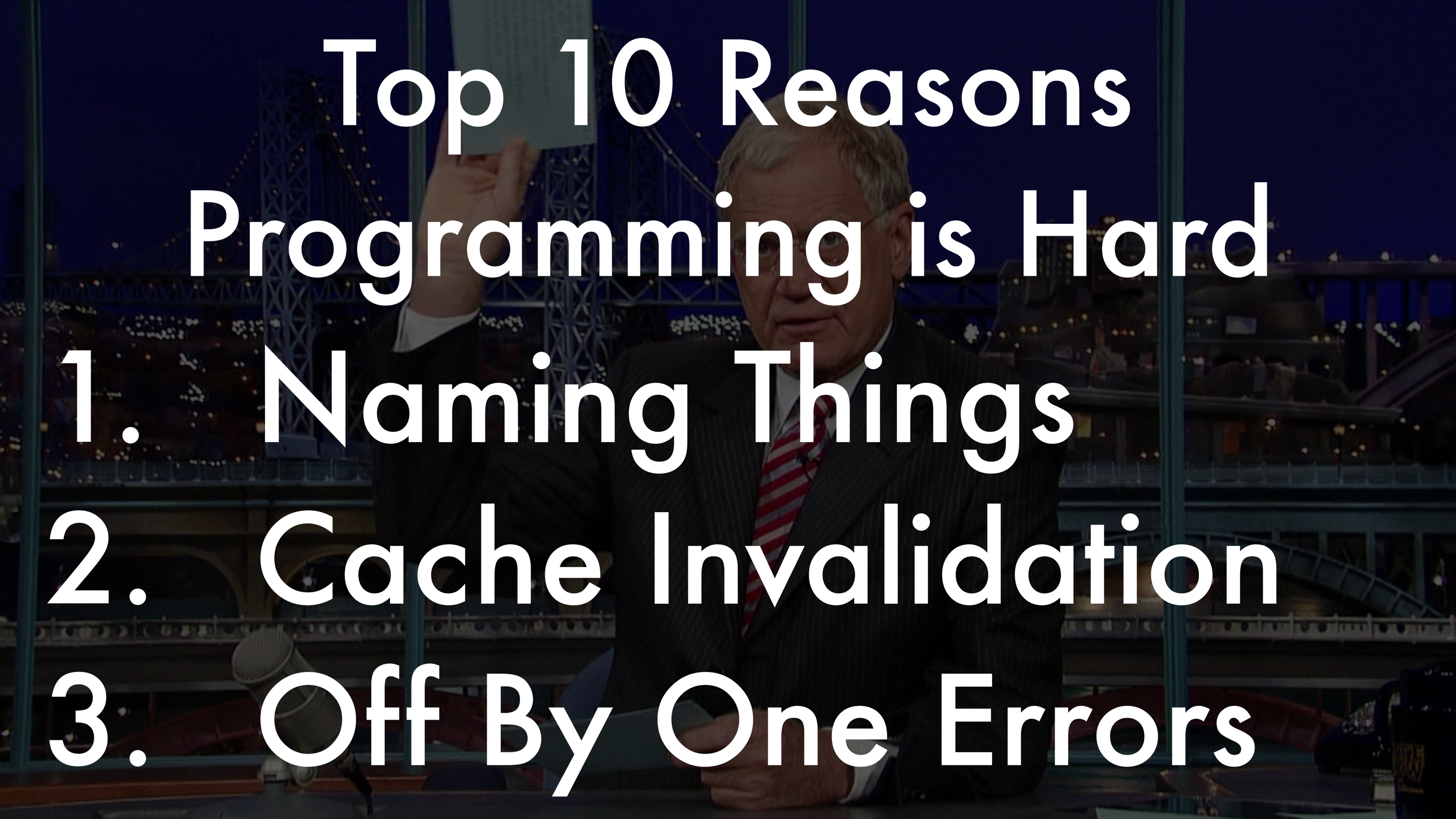
Ubiquitous Language

Customer

Client

Member

Affiliate



Top 10 Reasons Programming is Hard

1. Naming Things
2. Cache Invalidation
3. Off By One Errors

DDD	CRUD/MUD
Members	Users
LibraryCatalog	BookRepository
ISBN	String
DeweyDecimal	Float(7,3)

```
$catalog = new LibraryCatalog(…);
$book = $catalog->findOneBookByISBN($isbn);

$members = new Members(…);
$borrower = $members->findOneByAccountNumber($id);

$borrower->borrow($book);

$members->updateAccount($borrower);
```

```
$user = User::findOrFail($uuid);

$book = Books::where('isbn', '=', $isbn)->take(1)->get();

if (!$user->userBookLinks()->contains($book->id) && $user->borrowLinks->count() < 5) {

    $user->userBookLinks()->attach($book->id, ['date_due' => new DateTime('+2 Weeks')];

}

$user->save();
```

```
$catalog = new LibraryCatalog(…);
$book = $catalog->findOneBookByISBN($isbn);

$members = new Members(…);
$borrower = $members->findOneByAccountNumber($id);

$borrower->borrow($book)
```

```
$borrower->borrow($book);
$borrower->loan($book);
$borrower->obtain($book);
$borrower->take($book);
```

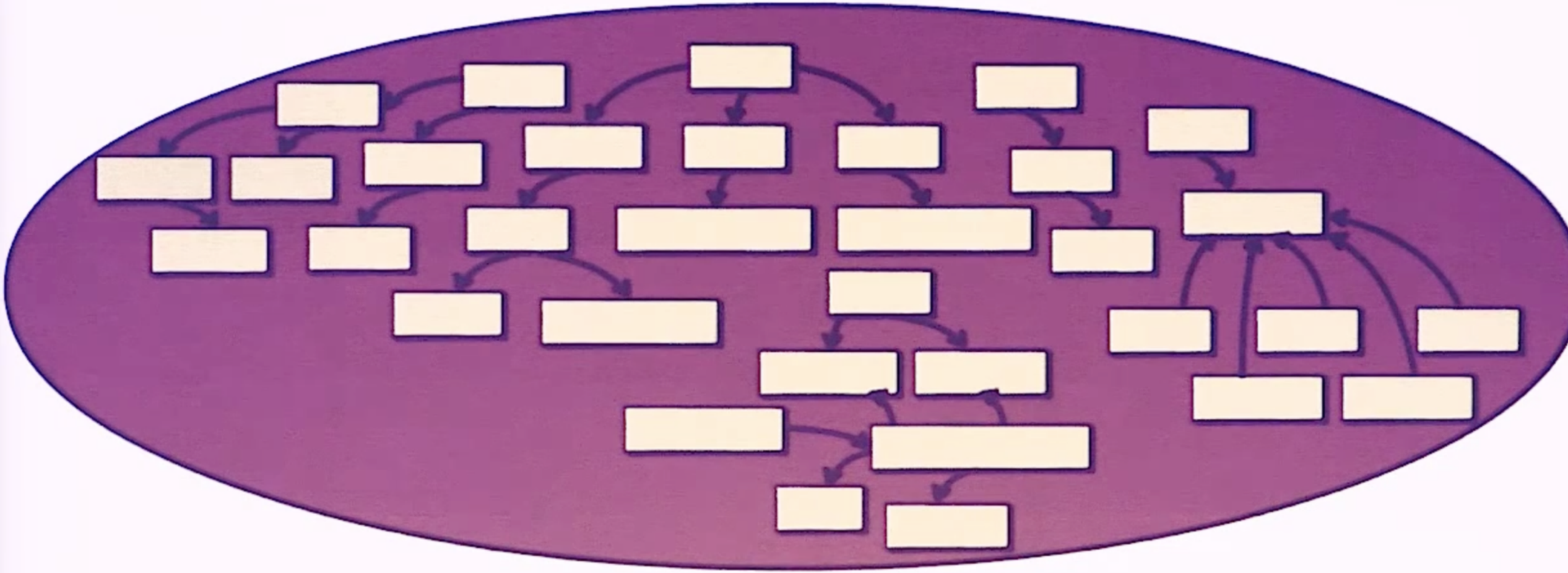
	DDD	BALL OF MUD/CRUD
Correctness	✓	
Testability		
Usability		
Maintainability		
Modifiability		

3D printing
workshops
mon 3-5 p.m.
THE BLACK KEYS
MADISON SQUARE GARDEN
SHRINE AUDITORIUM
SAT., DEC. 28th
AN UNFORGETTABLE MUSICAL EXPERIENCE!
...IN CONCERT
fischer
fischer

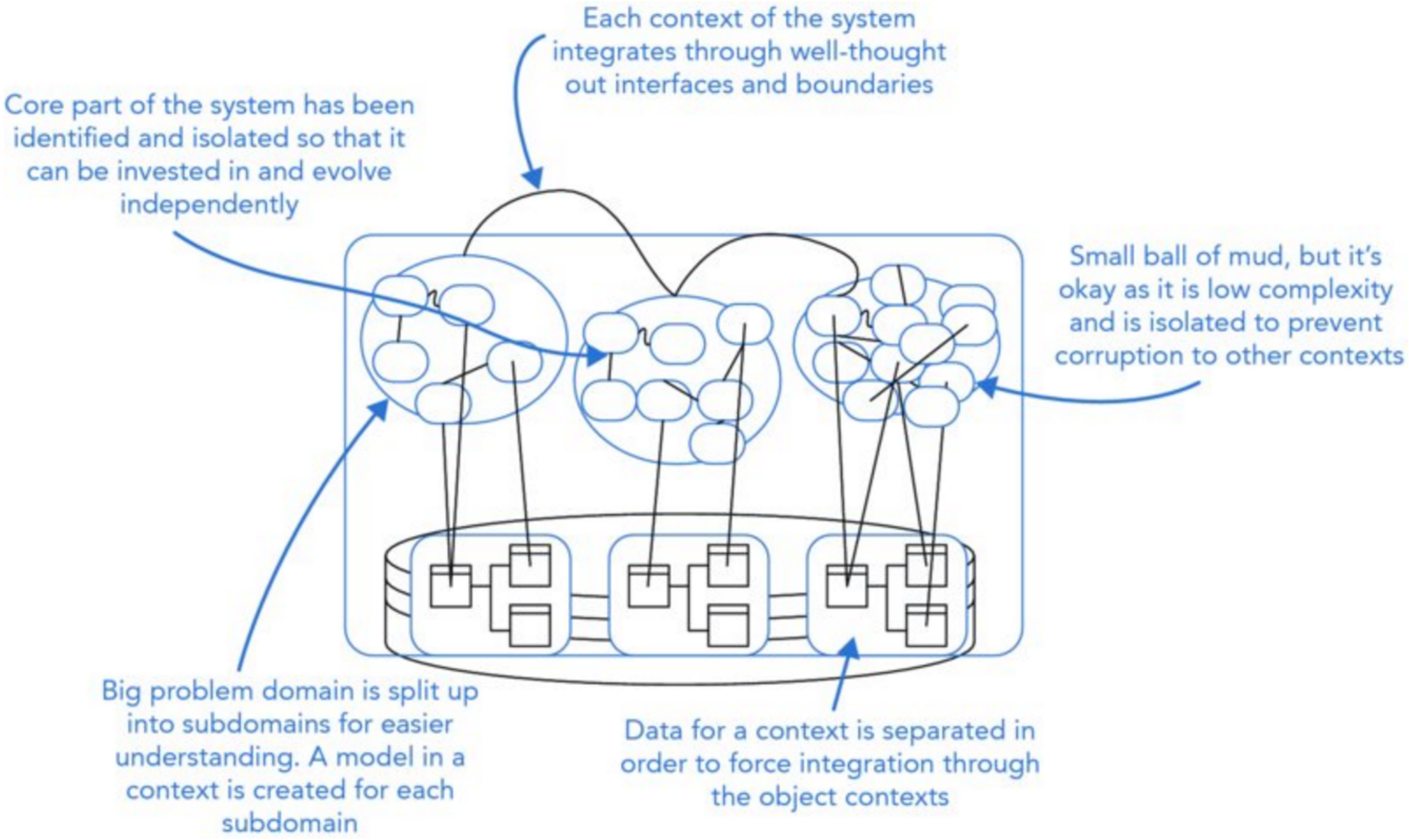
Production
Finance
Marketing
Sales

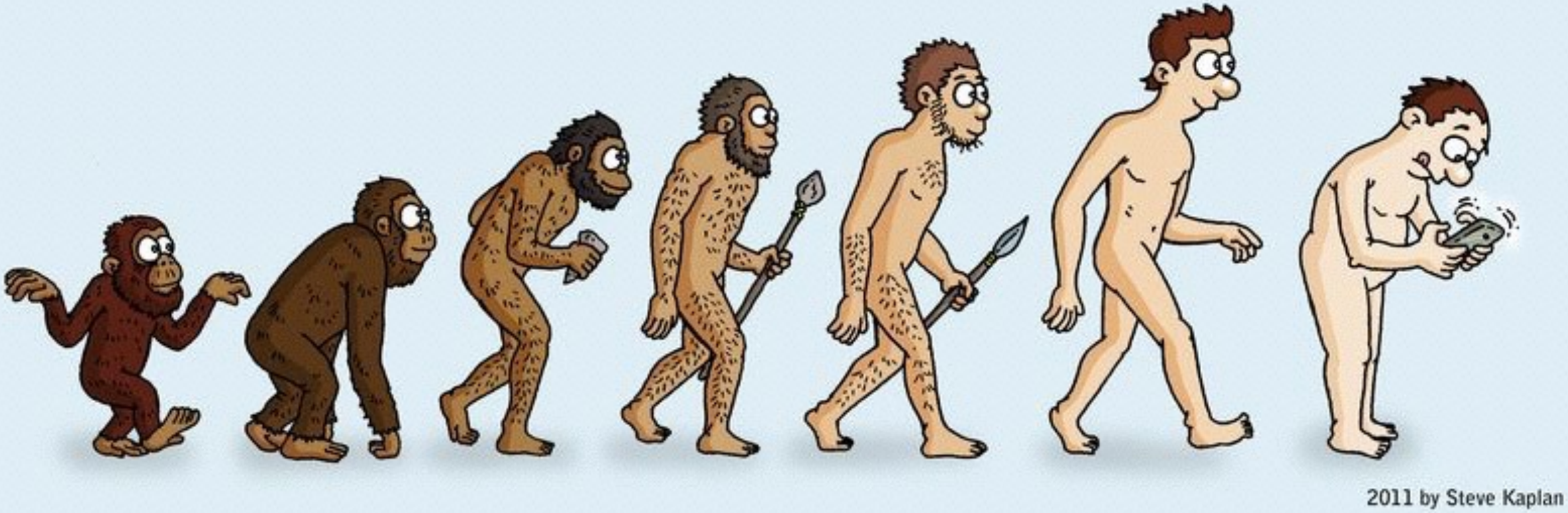
ShipTo
Payee
Lead
Customer

Big Ball of Mud

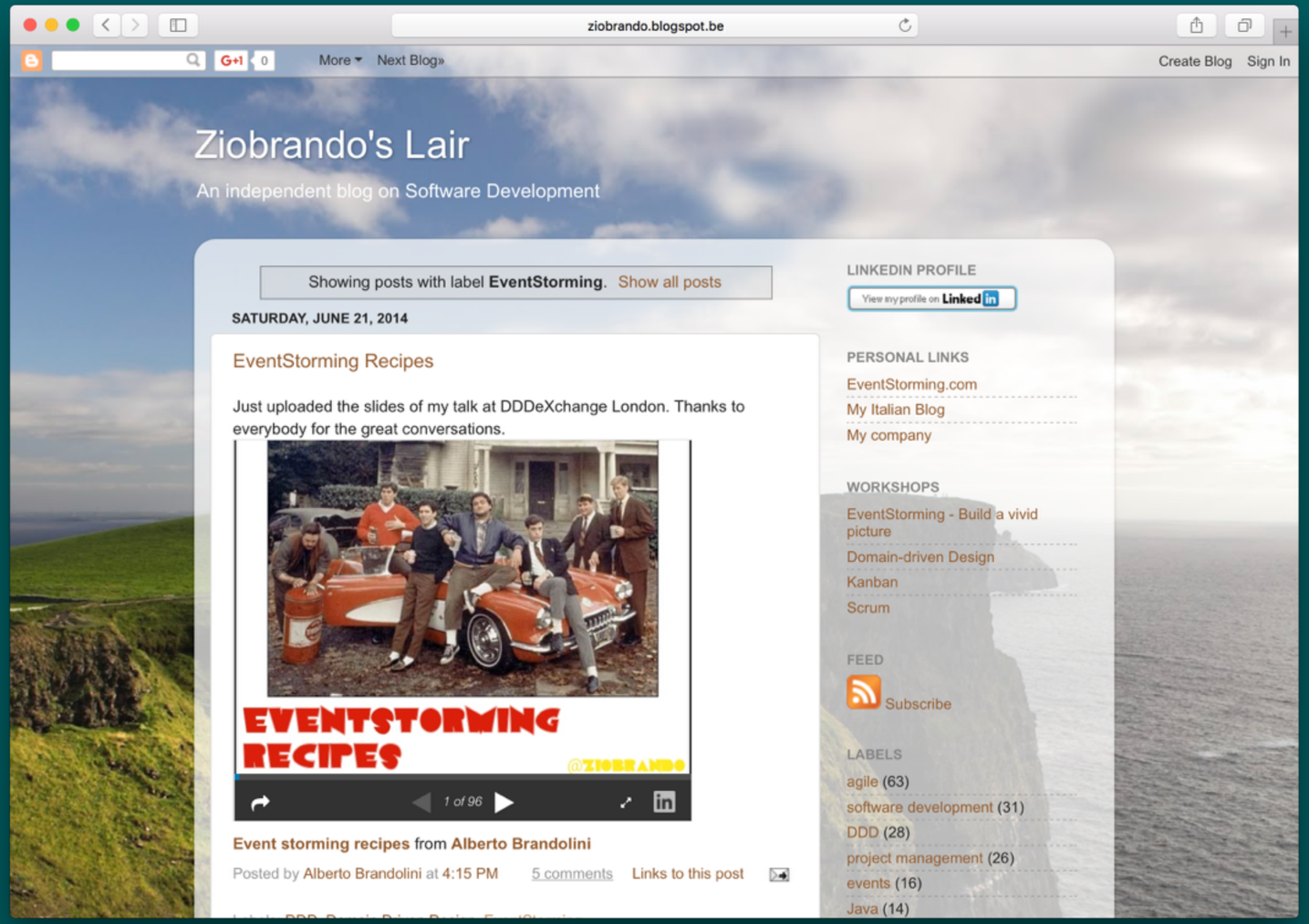


Slide: Vaughn Vernon


Each context of the system integrates through well-thought out interfaces and boundaries
Core part of the system has been identified and isolated so that it can be invested in and evolve independently
Small ball of mud, but it's okay as it is low complexity and is isolated to prevent corruption to other contexts
Big problem domain is split up into subdomains for easier understanding. A model in a context is created for each subdomain
Data for a context is separated in order to force integration through the object contexts


2011 by Steve Kaplan

Alberto Brandolini



http://ziobrando.blogspot.be/

Domain-Driven Design courses

with Mathias Verraes

Workshop formats

Event Storming

- One day hands-on discovery, analysis, modelling, and visualization

JIRA

Goat User Stories
@GoatUserStories

As a goat, I want to implement a RESTful interface

11:56 AM - 14 May 2015

7 Dirty Words When Meeting With a Domain Expert

1. Session
2. Repository
3. Abstract
4. Interface
5. Class
6. Database
7. Foreign Key

ftware craftsman with more than twenty-five years of experience in software design, . He is a thought leader in simplifying software design and implementation using en programming with object-oriented languages since the 1980s, and applying esign since his Smalltalk domain modeling days in the early 1990s. He consults and s taught his Implementing Domain-Driven Design classes on multiple continents.

45 DAYS FREE
ACCESS TO ONLINE EDITION
with purchase of this book
Details on Last Page

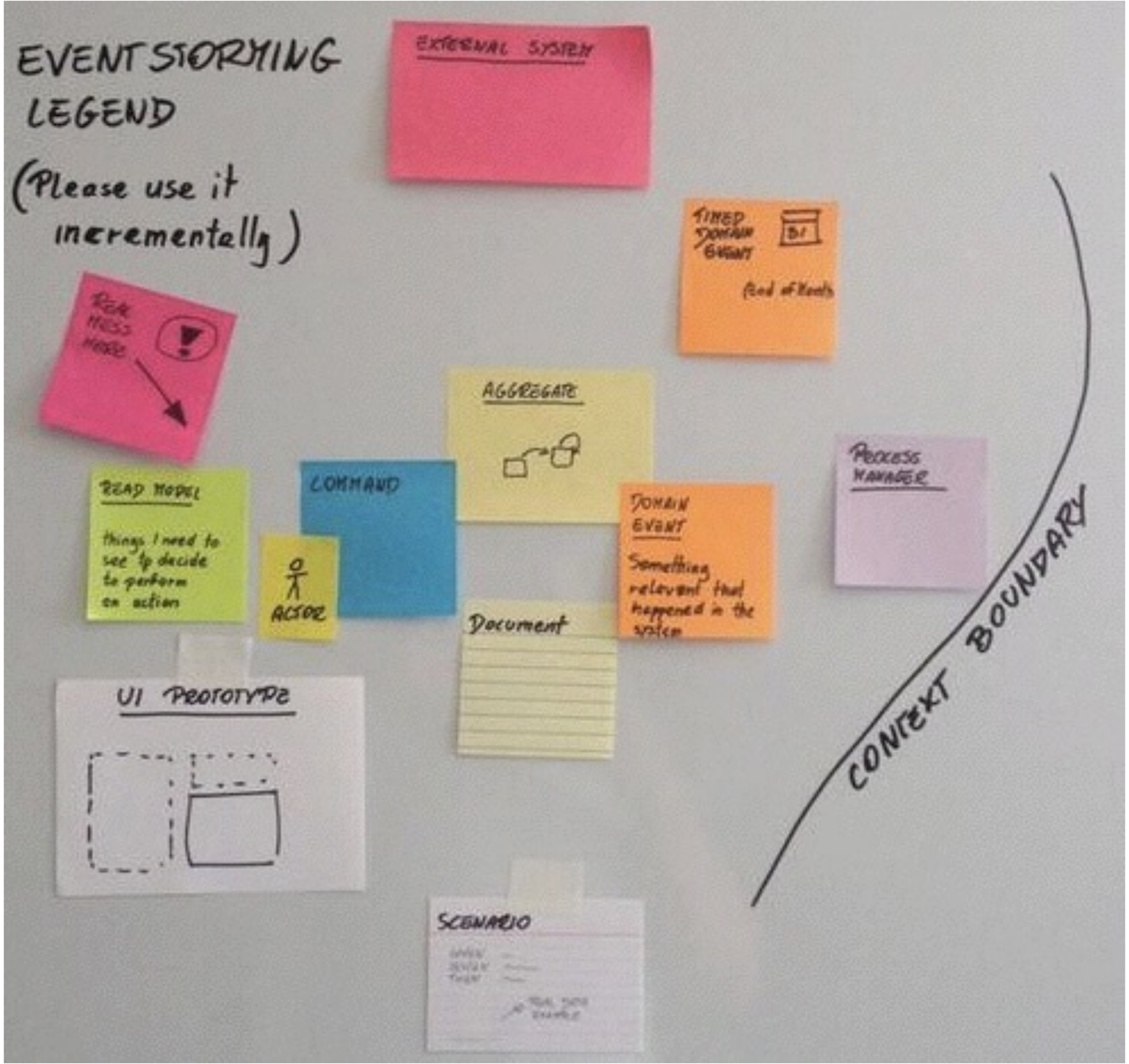
ISBN-13: 978-0-321-83457-7
ISBN-10: 0-321-83457-7

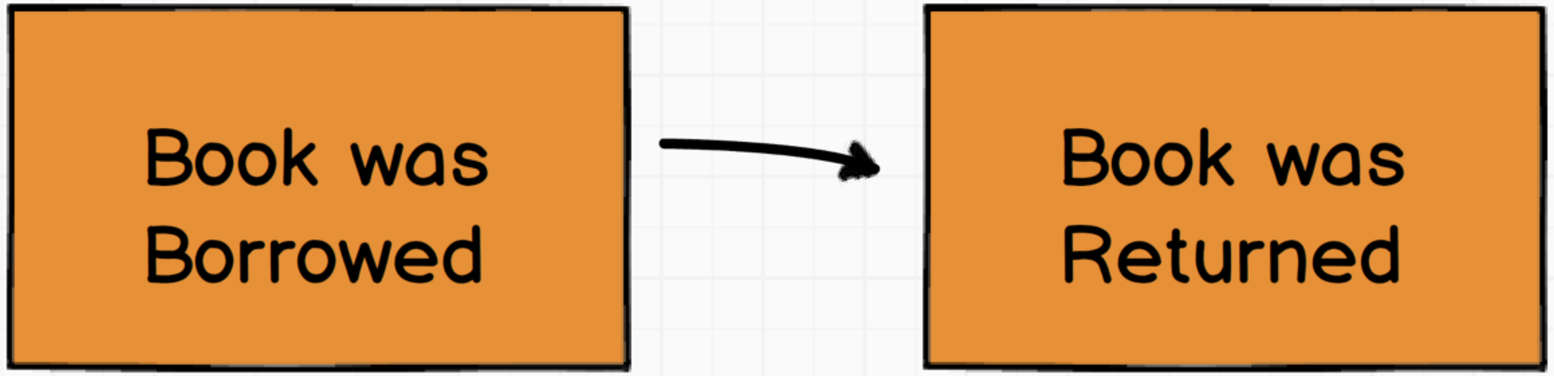
9 780321 834577 55499

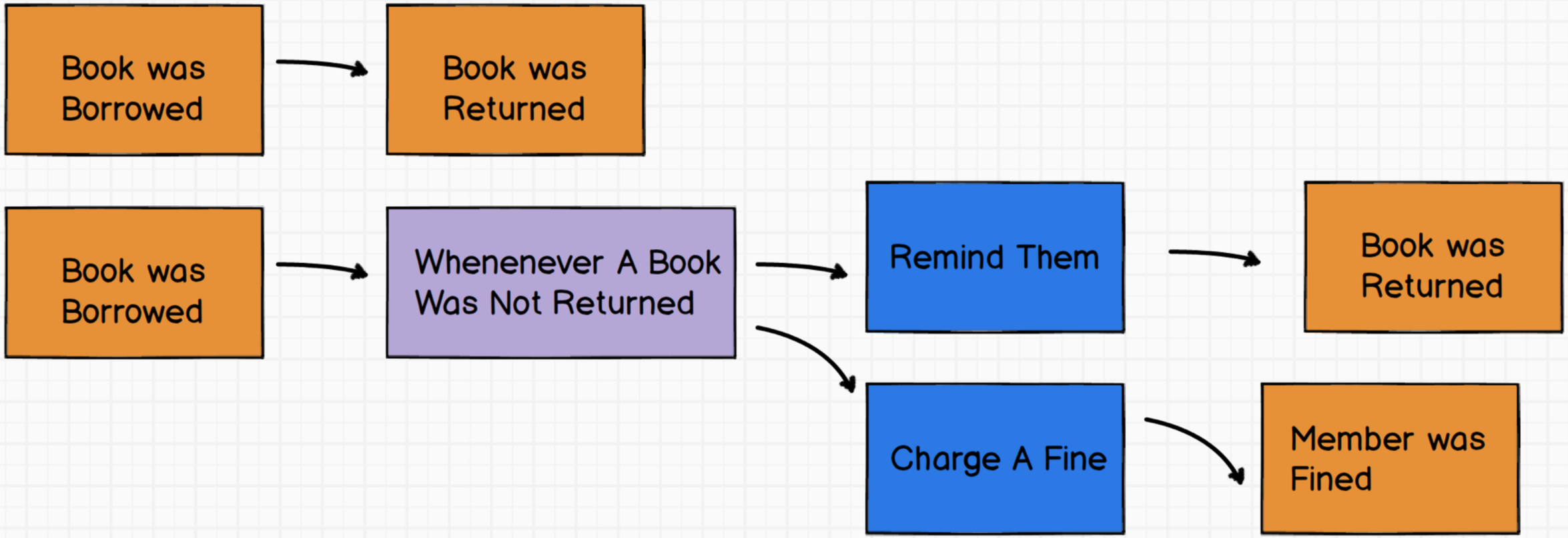
$54.99 US | $57.99 CANADA

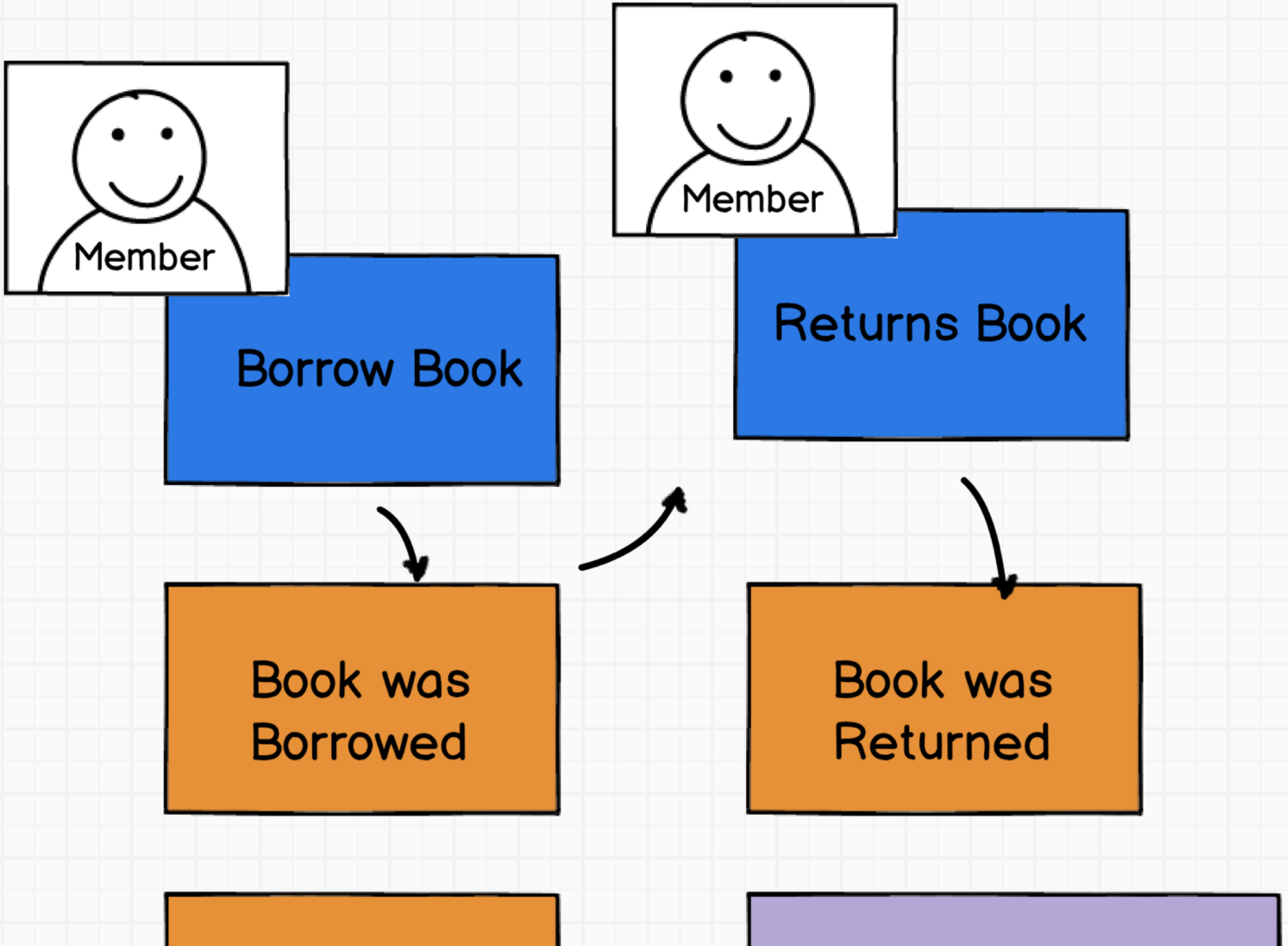
PEARSON

Photo: Mathias Verraes


EVENT STORMING
LEGEND
(Please use it
incrementally)
EXTERNAL SYSTEM
TIMED DOMAIN EVENT
BI
End of Month
AGGREGATE
PROCESS MANAGER
READ MODEL
things I need to see to decide to perform an action
COMMAND
ACTOR
DOMAIN EVENT
Something relevant that happened in the system
Document
UI PROTOTYPE
SCENARIO
CONTEXT BOUNDARY


Book was Borrowed
Book was Returned

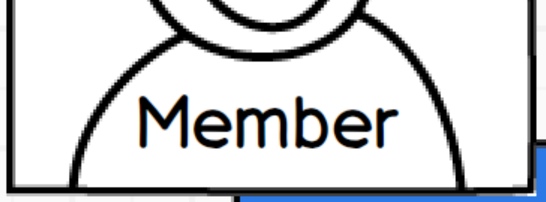

Book was Borrowed
Book was Returned
Book was Borrowed
Whenenever A Book Was Not Returned
Remind Them
Book was Returned
Charge A Fine
Member was Fined


Member
Borrow Book
Member
Returns Book
Book was Borrowed
Book was Returned

Borrow Book

Book was Borrowed

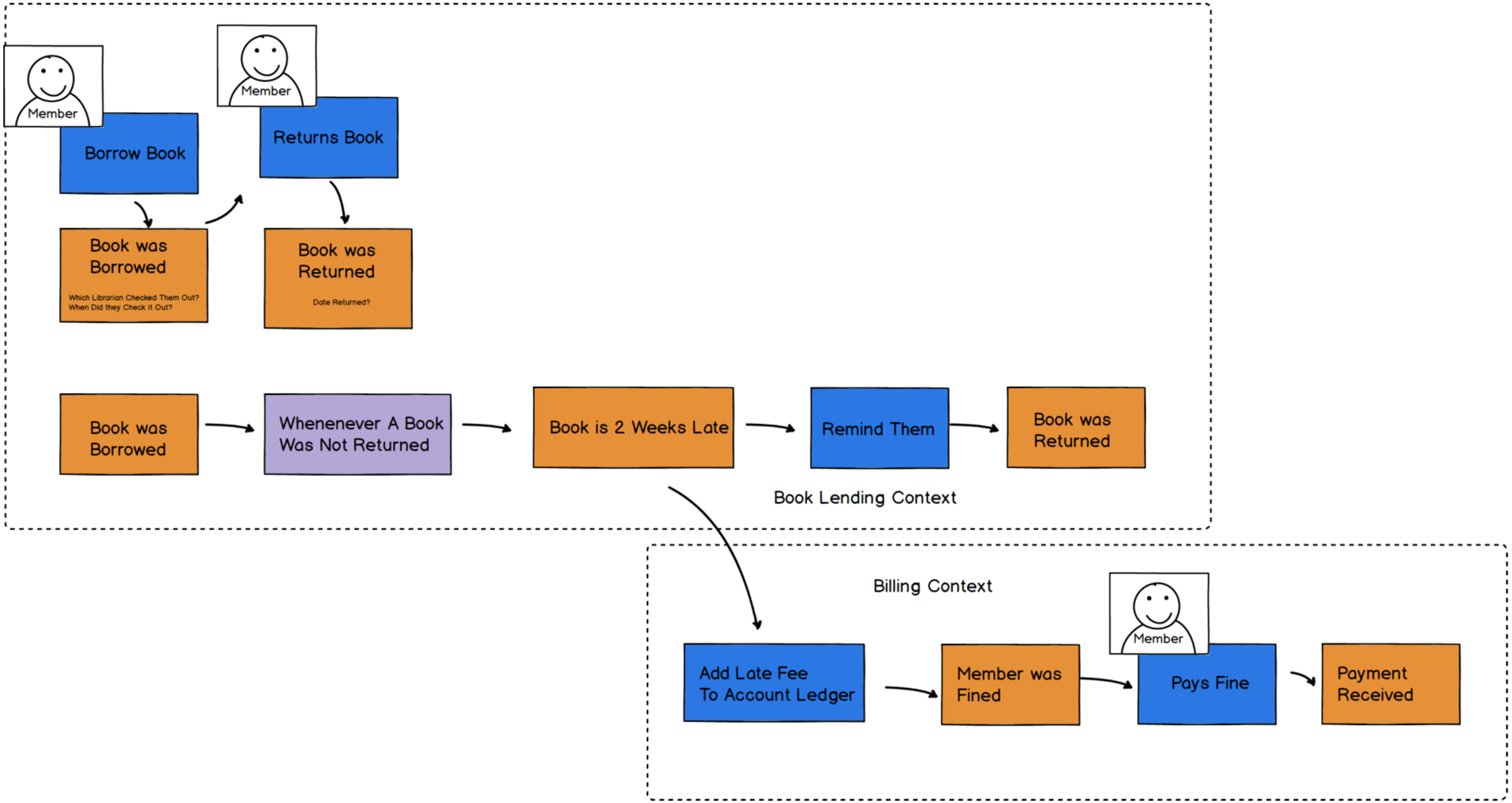
Which Librarian Checked Them Out?
When Did they Check it Out?

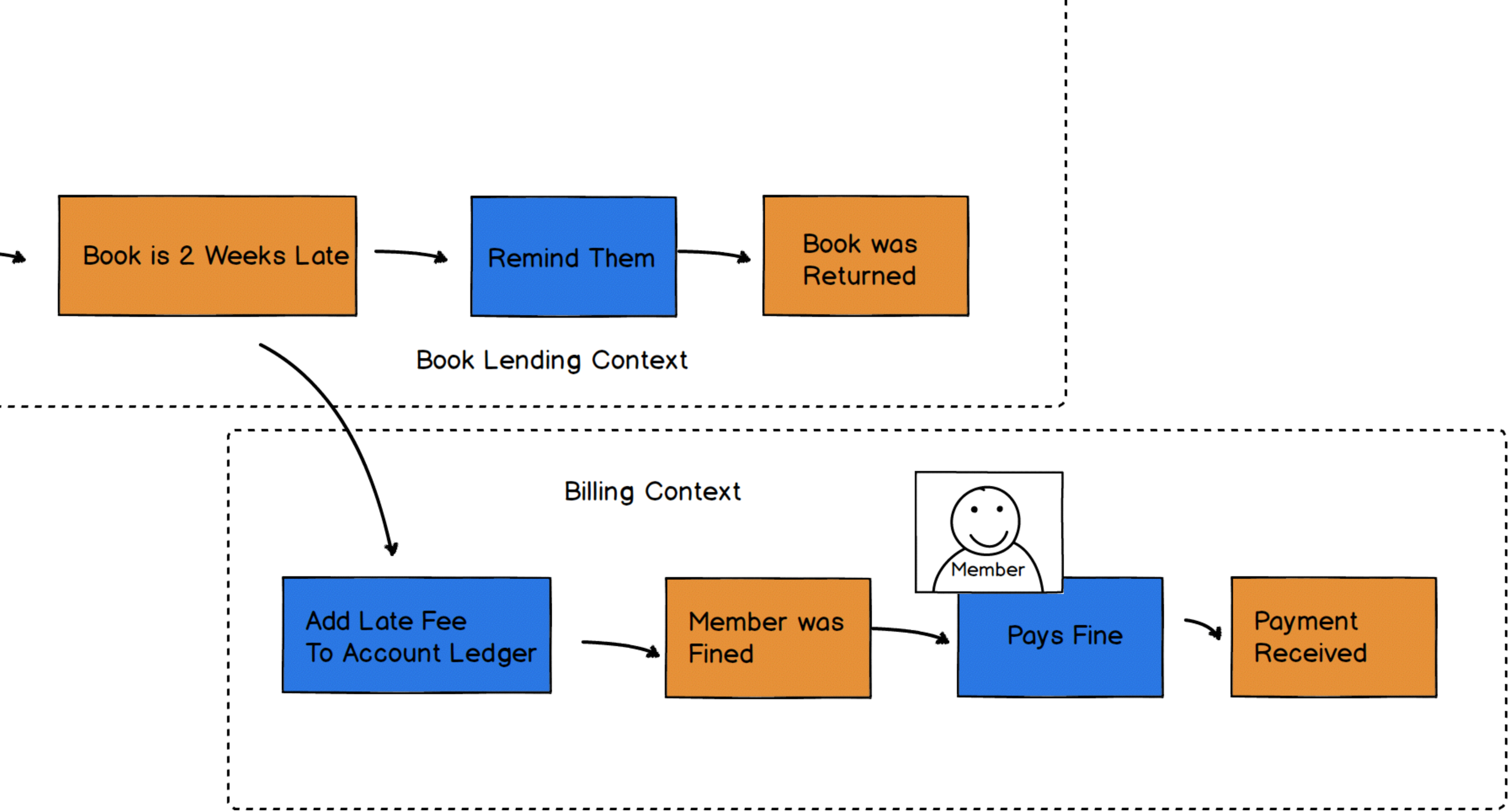


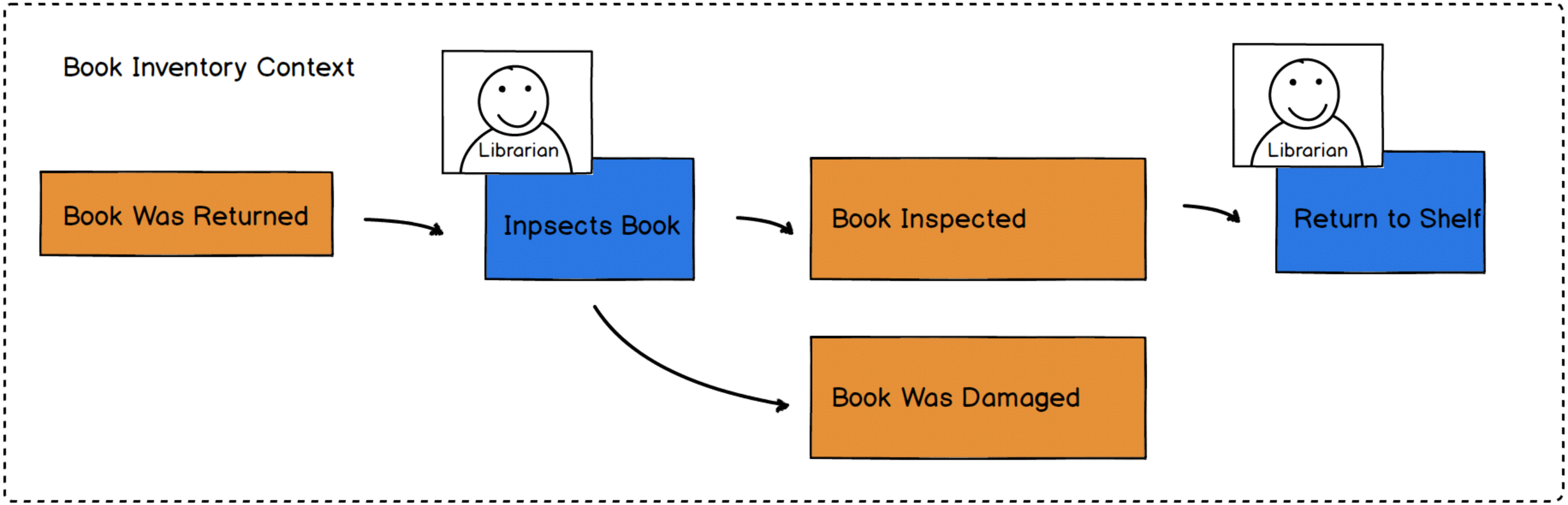
Returns Book

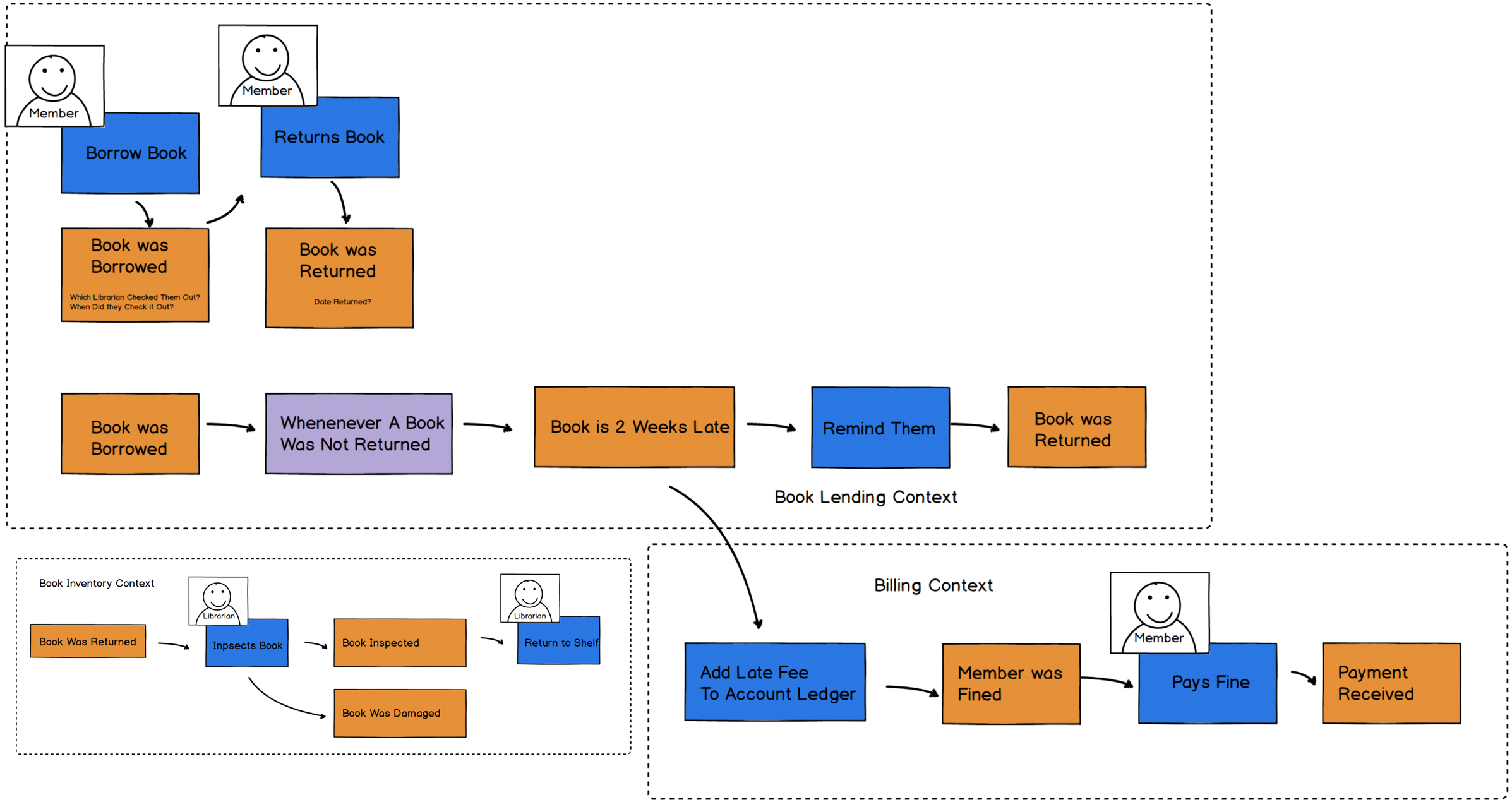
Book was Returned

Date Returned?

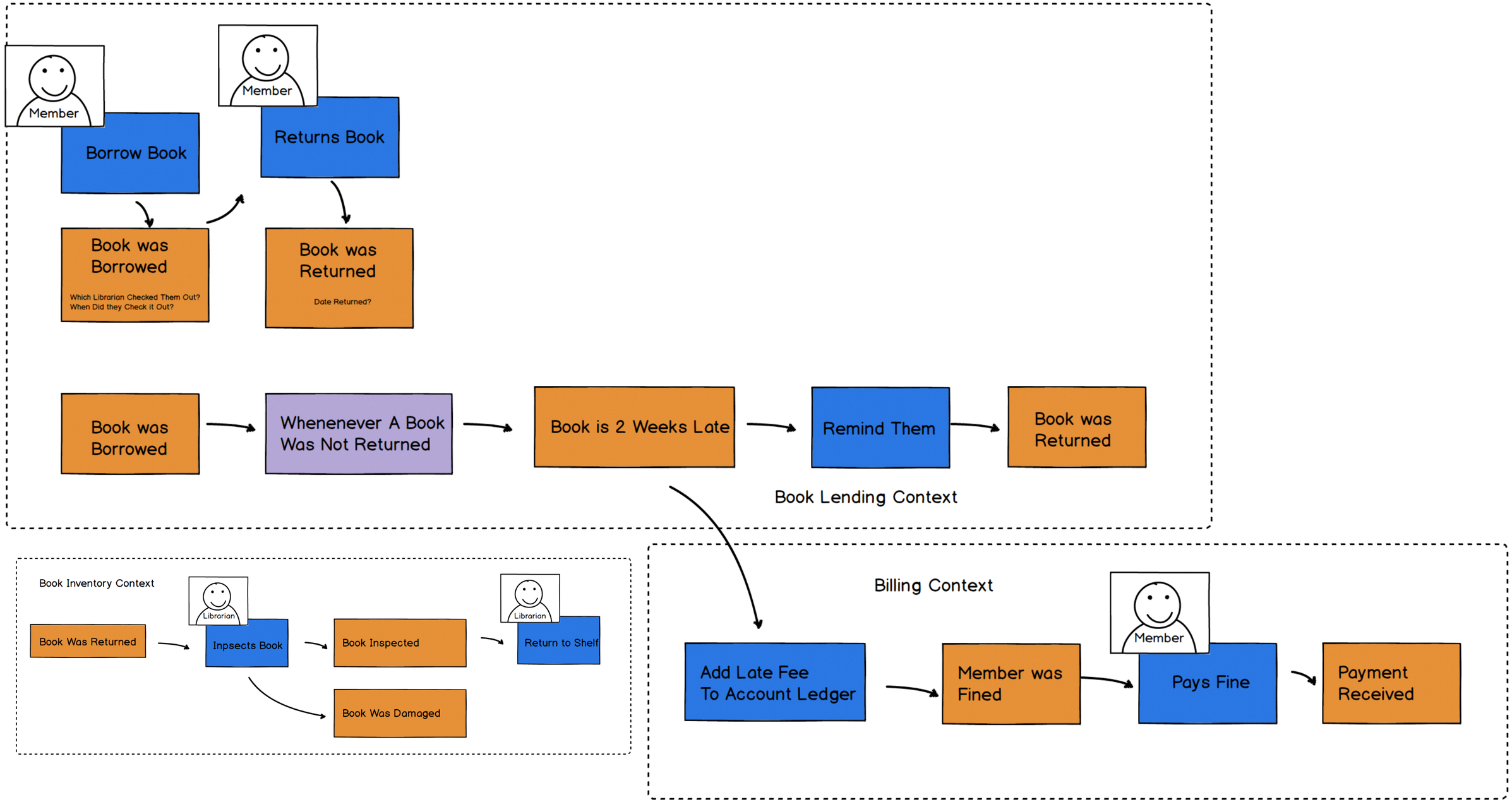

Member
Borrow Book
Member
Returns Book
Book was Borrowed
Which Librarian Checked Them Out?
When Did they Check it Out?
Book was Returned
Date Returned?
Book was Borrowed
Whenenever A Book Was Not Returned
Book is 2 Weeks Late
Remind Them
Book was Returned
Book Lending Context
Billing Context
Member
Add Late Fee To Account Ledger
Member was Fined
Pays Fine
Payment Received


Book is 2 Weeks Late
Remind Them
Book was Returned
Book Lending Context
Billing Context
Member
Add Late Fee To Account Ledger
Member was Fined
Pays Fine
Payment Received


Book Inventory Context
Book Was Returned
Librarian
Inpsects Book
Book Inspected
Book Was Damaged
Librarian
Return to Shelf


Member
Borrow Book
Member
Returns Book
Book was Borrowed
Which Librarian Checked Them Out?
When Did they Check it Out?
Book was Returned
Date Returned?
Book was Borrowed
Whenenever A Book Was Not Returned
Book is 2 Weeks Late
Remind Them
Book was Returned
Book Lending Context
Book Inventory Context
Librarian
Book Was Returned
Inpsects Book
Book Inspected
Librarian
Return to Shelf
Book Was Damaged
Billing Context
Add Late Fee To Account Ledger
Member was Fined
Member
Pays Fine
Payment Received

Why is the rum always gone?


Member
Borrow Book
Member
Returns Book
Book was Borrowed
Which Librarian Checked Them Out?
When Did they Check it Out?
Book was Returned
Date Returned?
Book was Borrowed
Whenenever A Book Was Not Returned
Book is 2 Weeks Late
Remind Them
Book was Returned
Book Lending Context
Book Inventory Context
Librarian
Book Was Returned
Inpsects Book
Book Inspected
Librarian
Return to Shelf
Book Was Damaged
Billing Context
Add Late Fee To Account Ledger
Member was Fined
Member
Pays Fine
Payment Received

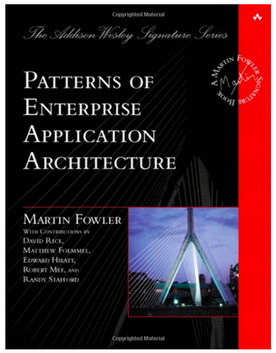
Domain Objects

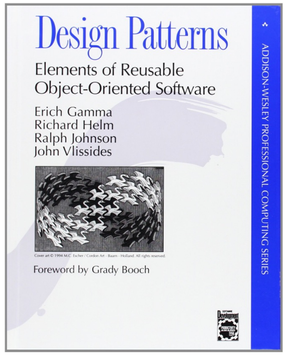
Repositories
Commands & Command handlers
Business rules/specifications
Messages
Domain events

Aggregates
Entities
Value Objects
Invariants
Processes

Domain Objects

Repositories
Commands & Command handlers
Business rules/specifications
Messages
Domain events





Aggregates
Entities
Value Objects
Invariants
Processes

Value Object	Immutable No Identity only Values
Entity	Mutable Has A Lifecycle Always Stable Contains Value Objects
Aggregate	Entity Responsible For Child Entities

Value Objects

Amount Paid
Email Address
BookTitle
Dewey Decimal Number
ISBN Number
Domain Event

Value Objects

Mathias Verraes 10/31/14

VO's unless you have a really good reason not to.

- show quoted text -

- show quoted text -

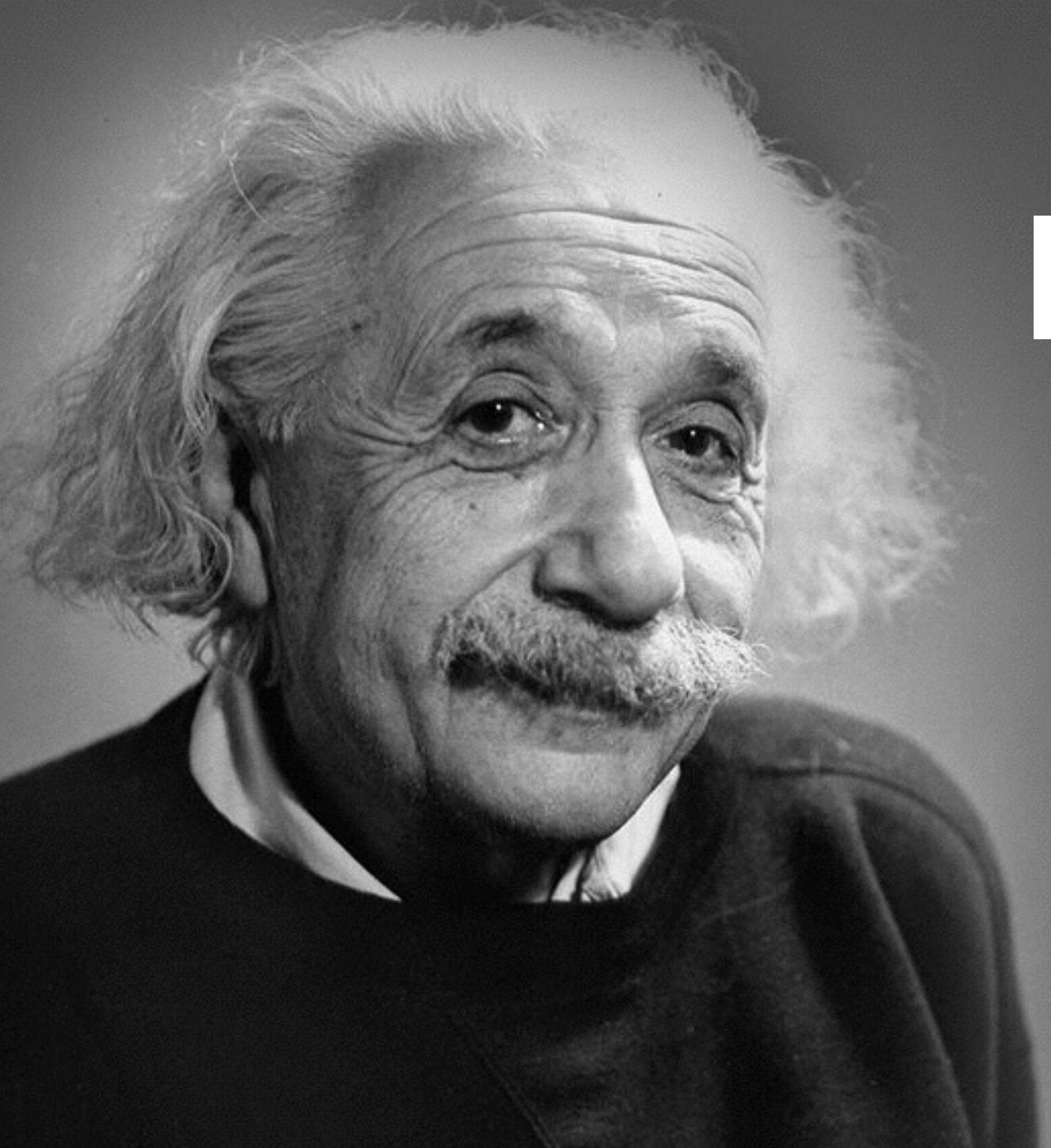
To view this discussion on the web visit https://groups.google.com/d/msgid/dddinphp/2f743db2-6669-4f7d-835b-e9545e21debb%40googlegroups.com.

For more options, visit https://groups.google.com/d/optout.

Mathias Verraes 10/31/14

An optimisation is not premature if you know the outcome in advance. VOs attract behaviour. Without VOs, you blind yourself for opportunities to encapsulate. <argument from authority>Eric Evans agrees with me</ argument from authority>

- show quoted text -



Immutability

Avoid Spooky Action at a Distance

Value Objects

Value Objects

Not Just Static Typing

https://github.com/Fiedzia/type-system-research/blob/master/README.md

Value Objects

"Gateway Drug to
Test Driven Development"

#MYTESTSDONTPASS

Value Objects

Plain PHP Objects (POPOs) That are Immutable

- Declare Class Properties as Private
- No Setters
- All Parameters Are Injected in the Constructor
- Do Not Store References to Mutable Objects (Private Scalars Okay)
- Functions that Appear to Modify Should Return a New Object
- Declare the Class Final so it can't be Overridden (probably)

Example

```
BookTitle
```

```php
class BookTitleTest extends \PHPUnit_Framework_TestCase
{
    /**
     * @test
     */
    public function testBookTitle()
    {
        $bookTitle = new BookTitle("The Lion, the Witch and the Wardrobe");
        $this->assertEquals("The Lion, the Witch and the Wardrobe",$bookTitle->getValue());
    }

    /**
     * @test
     * @expectedException \Exception
     */
    public function testEmptyBookTitle()
    {
        new BookTitle("");
    }

}
```

```
[vagrant@localhost library]$ vendor/bin/phpunit
PHPUnit 4.8.26 by Sebastian Bergmann and contributors.

PHP Fatal error:  Call to undefined method Cassell\Library\Book\BookTitle::getValue() in /srv/www/presen
tations/ddd/library/tests/Book/BookTitleTest.php on line 14
PHP Stack trace:
PHP   1. {main}() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/phpunit:0
PHP   2. PHPUnit_TextUI_Command::main() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/phpuni
t:47
PHP   3. PHPUnit_TextUI_Command->run() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/src/Tex
tUI/Command.php:100
PHP   4. PHPUnit_TextUI_TestRunner->doRun() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/sr
c/TextUI/Command.php:149
PHP   5. PHPUnit_Framework_TestSuite->run() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/sr
c/TextUI/TestRunner.php:440
PHP   6. PHPUnit_Framework_TestSuite->run() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/sr
c/Framework/TestSuite.php:747
PHP   7. PHPUnit_Framework_TestCase->run() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/src
/Framework/TestSuite.php:747
PHP   8. PHPUnit_Framework_TestResult->run() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/s
rc/Framework/TestCase.php:724
PHP   9. PHPUnit_Framework_TestCase->runBare() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit
/src/Framework/TestResult.php:612
PHP  10. PHPUnit_Framework_TestCase->runTest() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit
/src/Framework/TestCase.php:768
PHP  11. ReflectionMethod->invokeArgs() /srv/www/presentations/ddd/library/vendor/phpunit/phpunit/src/Fr
amework/TestCase.php:909
PHP  12. Cassell\Library\Tests\Book\BookTitleTest->testBookTitle() /srv/www/presentations/ddd/library/ve
ndor/phpunit/phpunit/src/Framework/TestCase.php:909

Fatal error: Call to undefined method Cassell\Library\Book\BookTitle::getValue() in /srv/www/presentatio
ns/ddd/library/tests/Book/BookTitleTest.php on line 14
```

```php
namespace Cassell\Library\Book;

class BookTitle
{
    private $titleString;

    public function __construct($titleString)
    {
        $this->titleString = $titleString;
    }

    /**
     * @return mixed
     */
    public function getValue()
    {
        return $this->titleString;
    }

}
```

```
[vagrant@localhost library]$ vendor/bin/phpunit
PHPUnit 4.8.26 by Sebastian Bergmann and contributors.

.F

Time: 273 ms, Memory: 4.75MB

There was 1 failure:

1) Cassell\Library\Tests\Book\BookTitleTest::testEmptyBookTitle
Failed asserting that exception of type "\Exception" is thrown.

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
```

```php
namespace Cassell\Library\Book;

class BookTitle
{
    private $titleString;

    public function __construct($titleString)
    {
        if (empty($titleString)) {
            throw new \Exception("Title can not be empty");
        }
        $this->titleString = $titleString;
    }

    /**
     * @return mixed
     */
    public function getValue()
    {
        return $this->titleString;
    }

}
```

```
[vagrant@localhost library]$ vendor/bin/phpunit
PHPUnit 4.8.26 by Sebastian Bergmann and contributors.

..

Time: 258 ms, Memory: 4.75MB

OK (2 tests, 2 assertions)
```

LIGHT IS GREEN
TESTS ARE CLEAN
imgflip.com

Example

```
BookCondition
```

```php
class BookConditionTest  extends \PHPUnit_Framework_TestCase
{
    /**
     * @test
     */
    public function testBookConditionValues()
    {
        $likeNew = new BookCondition("LIKE NEW");
        $moderate = new BookCondition("MODERATELY WORN");
        $heavy = new BookCondition("HEAVILY WORN");
        $damaged = new BookCondition("DAMAGED");

        $this->assertEquals("LIKE NEW",$likeNew->getValue());
        $this->assertEquals("MODERATELY WORN",$moderate->getValue());
        $this->assertEquals("HEAVILY WORN",$heavy->getValue());
        $this->assertEquals("DAMAGED",$damaged->getValue());
    }

    /**
     * @test
     * @expectedException \Exception
     */
    public function testInvalidBookCondition()
    {
        new BookCondition("Apple");
    }

    /**
     * @test
     */
    public function testDegraded()
    {
        $likeNew = new BookCondition("LIKE NEW");
        $degraded = $likeNew->degrade();
        $this->assertEquals("MODERATELY WORN",$degraded->getValue());
    }

    /**
     * @test
     * @expectedException \Exception
     */
    public function testCantDegradeDamaged()
    {
        $damaged = new BookCondition("DAMAGED");
```

```php
/**
 * @test
 */
public function testDegraded()
{
    $likeNew = new BookCondition("LIKE NEW");
    $degraded = $likeNew->degrade();
    $this->assertEquals("MODERATELY WORN",$degraded->getValue());
}

/**
 * @test
 * @expectedException \Exception
 */
public function testCantDegradeDamaged()
{
    $damaged = new BookCondition("DAMAGED");
    $damaged->degrade();
}
```

```php
class BookCondition
{
    const LIKE_NEW = "LIKE NEW";
    const MODERATELY_WORN = "MODERATELY WORN";
    const HEAVILY_WORN = "HEAVILY WORN";
    const DAMAGED = "DAMAGED";

    private $condition;

    public function __construct($condition)
    {
        if(!in_array($condition, $this->getPossibleConditions())) {
            throw new \Exception("Invalid Book Condition Type");

        }
        $this->condition = $condition;
    }

    public function getValue()
    {
        return $this->condition;
    }

    public function degrade()
    {
        $currentArrayKey = array_search($this->getValue(),$this->getPossibleConditions());
        $nextKey = $currentArrayKey + 1;
        $nextCondition = $this->getPossibleConditions()[$nextKey];
        return new self($nextCondition);
    }

    public function getPossibleConditions()
    {
        return [self::LIKE_NEW, self::MODERATELY_WORN, self::HEAVILY_WORN, self::DAMAGED];
    }
```

```
        $this->condition = $condition;
    }

    public function getValue()
    {
        return $this->condition;
    }

    public function degrade()
    {
        $currentArrayKey = array_search($this->getValue(),$this->getPossibleCondit
        $nextKey = $currentArrayKey + 1;
        $nextCondition = $this->getPossibleConditions()[$nextKey];
        return new self($nextCondition);
    }

    public function getPossibleConditions()
    {
        return [self::LIKE_NEW, self::MODERATELY_WORN, self::HEAVILY_WORN, self::D
    }

}
```

ISBN

DeweyDecimal

…

Example

```
DateBorrowed
```

```php
class DateBorrowed
{
    /**
     * @var \DateTimeImmutable
     */
    private $dateTime;

    public function __construct(\DateTimeImmutable $dateTime)
    {
        if ($dateTime < new \DateTimeImmutable("May 26 2016")) {
            throw new \Exception("Invalid Past Date");
        } else if ($dateTime > new \DateTimeImmutable("May 26 3016")) {
            throw new \Exception("Invalid Future Date ");
        }

        $this->dateTime = $dateTime;
    }
```

Example

```
LoanPeriod
```

```php
class LoanPeriod
{
    public function __construct(DateBorrowed $dateBorrowed, Renewals $renewals, DateReturned $dateReturned)
    {
        $this->dateBorrowed = $dateBorrowed;
        $this->dateReturned = $dateReturned;
        $this->renewals = $renewals;
    }

    public function getLengthInDays()
    {
        return $this->dateBorrowed->numberOfDaysUntil($this->dateReturned);
    }

}
```

DDD	CRUD/MUD
BookTitle	String
BookCondition	String
ISBN	String
DeweyDecimal	Float(7,3)
DateBorrowed	DateTime

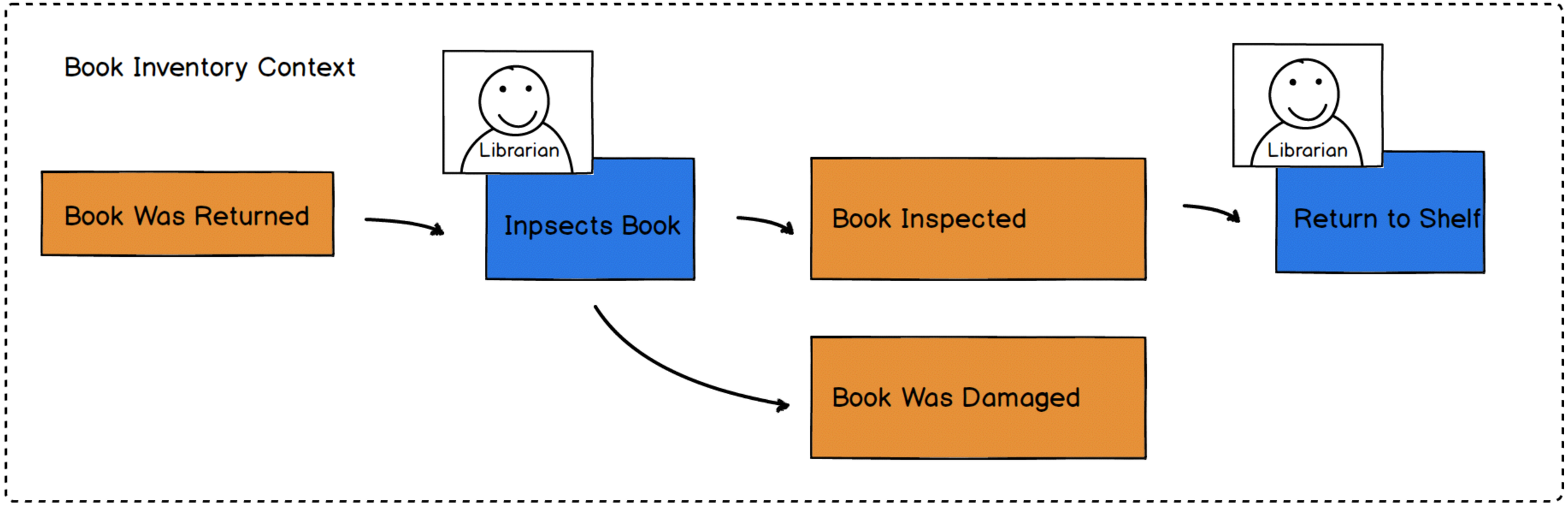
	DDD	BALL OF MUD/CRUD
Correctness	✓	
Testability	✓	
Usability		
Maintainability		
Modifiability		

Entities

Identifiable
Has State and is Mutable
Never Invalid State
Required Parameters Passed in Constructor
Operates Using Value Objects
No Security or Permission Checks
Storage Agnostic

Entities

Librarian
Book


Book Inventory Context
Book Was Returned
Librarian
Inpsects Book
Book Inspected
Book Was Damaged
Librarian
Return to Shelf

```php
class Book
{
    /**
     * @var BookCondition
     */
    private $bookCondition;
    /**
     * @var BookTitle
     */
    private $bookTitle;

    /**
     * @var BookTitle
     */
    private $bookId;

    public function __construct(BookTitle $bookTitle, BookCondition $bookCondition)
    {
        $this->bookId = new BookId(); // uuid
        $this->bookCondition = $bookCondition;
        $this->bookTitle = $bookTitle;
    }

    /**
     * @return BookCondition
     */
    public function getBookCondition()
    {
        return $this->bookCondition;
    }

    /**
     * @return BookTitle
     */
    public function getBookTitle()
    {
        return $this->bookTitle;
    }


}
```

```php
<?php

use Propel\Tests\Bookstore\Author;
use Propel\Tests\Bookstore\Book;
use Propel\Tests\Bookstore\BookQuery;
use Propel\Runtime\ActiveQuery\Criteria;

$book = new Book();
$book->setTitle('Lord of Propel');
$book->setPrice(25);

$author = new Author();
$author->setFirstName('Steve');

$book->setAuthor($author);
$book->save();

$books = BookQuery::create()
    ->filterByPrice(25, CRITERIA::GREATER_THAN)
    ->find();

foreach ($books as $book) {
    $book->setPrice(20);
}

$books->save();
```

SETTERS
ARE BAD

```php
class Book extends ActiveRecordEntity
{
    public function __construct($data)
    {
        parent::__construct($data);
        if($this->getBookTitle() == "") {
            throw  new \Exception("Invalid Book Title");
        }

    }
```

```php
class Book
{
    /**
     * @var BookCondition
     */
    private $bookCondition;
    /**
     * @var BookTitle
     */
    private $bookTitle;

    public function __construct(BookTitle $bookTitle, BookCondition $bookCondition)
    {
        $this->bookCondition = $bookCondition;
        $this->bookTitle = $bookTitle;
    }

    /**
     * @return BookCondition
     */
    public function getBookCondition()
    {
        return $this->bookCondition;
    }

    /**
     * @return BookTitle
     */
    public function getBookTitle()
    {
        return $this->bookTitle;
    }
```

```
    $this->bookTitle = $bookTitle;
}

/**
 * @return BookCondition
 */
public function getBookCondition()
{
    return $this->bookCondition;
}

/**
 * @return BookTitle
 */
public function getBookTitle()
{
    return $this->bookTitle;
}
```

```php
class OverdueBookNotice
{
    public function __construct($bookId, $bookTitle, $memberId)
    {
        if(empty($bookId)) {
            throw new \Exception("Book ID Required");
        }
        if(empty($bookTitle)) {
            throw new \Exception("Book Title Required");
        }
        if(empty($memberId)) {
            throw new \Exception("Member Required");
        }

        $this->bookId = $bookId;
        $this->bookTitle = $bookTitle;
        $this->memberId = $memberId;
    }
```

```php
class OverdueBookNotice
{
    public function __construct(BookId $bookId, BookTitle $bookTitle, MemberId $memberId)
    {
        $this->bookId = $bookId;
        $this->bookTitle = $bookTitle;
        $this->memberId = $memberId;
    }
```

```php
class Book
{
    /**
     * @var BookCondition
     */
    private $bookCondition;
    /**
     * @var BookTitle
     */
    private $bookTitle;

    /**
     * @var BookTitle
     */
    private $bookId;

    public function __construct(BookTitle $bookTitle, BookCondition $bookCondition)
    {
        $this->bookId = new BookId(); // uuid
        $this->bookCondition = $bookCondition;
        $this->bookTitle = $bookTitle;
    }
```

Doctrine

```php
/**
 * @Entity
 * @Table(name="book")
 */
class Book
{
    /**
     * @Id
     * @Column(type="string", name="book_condition")
     */
    private $bookCondition;

    /**
     * @return BookCondition
     */
    public function getBookTitle()
    {
        return new BookCondition($this->bookCondition);
    }
```

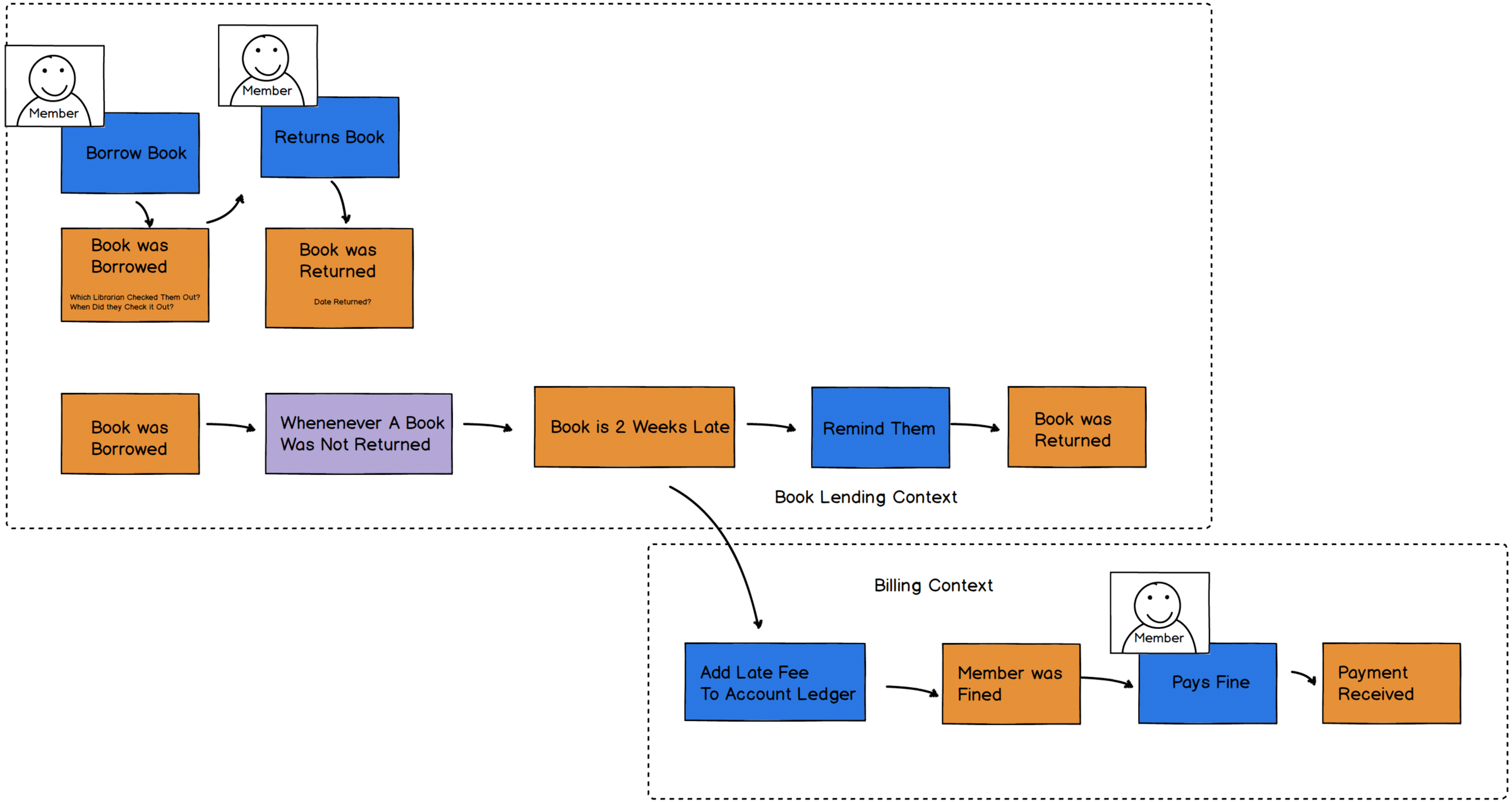
Aggregates

- Special Kind of Entity
- Manages Child Entities
- Transactionally Safe

Aggregates

Member

Fines Invoice


Member
Borrow Book
Member
Returns Book
Book was Borrowed
Which Librarian Checked Them Out?
When Did they Check it Out?
Book was Returned
Date Returned?
Book was Borrowed
Whenenever A Book Was Not Returned
Book is 2 Weeks Late
Remind Them
Book was Returned
Book Lending Context
Billing Context
Member
Add Late Fee To Account Ledger
Member was Fined
Pays Fine
Payment Received

```php
class Member
{
    // ...

    /**
     * @ManyToMany(targetEntity="FineAmount")
     * @JoinTable(name="member_fine_paid",
     *      joinColumns={@JoinColumn(name="fine_id", referencedColumnName="fine_id")},
     *      inverseJoinColumns={@JoinColumn(name="member_id", referencedColumnName="member_id", unique=true)}
     * )
     */
    private $payments;

    public function payFine(FineAmount $fineAmount)
    {
        $this->payments[] = $fineAmount;
    }

}
```

```php
class Member
{
    // ...

    /**
     * @ManyToMany(targetEntity="Book")
     * @JoinTable(name="member_loaned_book",
     *      joinColumns={@JoinColumn(name="book_id", referencedColumnName="book_id")},
     *      inverseJoinColumns={@JoinColumn(name="member_id", referencedColumnName="member_id", unique=true)}
     * )
     */
    private $booksOnLoan;

    public function borrow(Book $book)
    {
        $this->booksOnLoan[] = $book;
    }
```

```php
class Member
{
    // ...
    const BOOK_LIMIT = 5;

    /**
     * @ManyToMany(targetEntity="Book")
     * @JoinTable(name="member_loaned_book",
     *      joinColumns={@JoinColumn(name="book_id", referencedColumnName="book_id")},
     *      inverseJoinColumns={@JoinColumn(name="member_id", referencedColumnName="member_id", unique=true)}
     * )
     */
    private $booksOnLoan;

    public function borrow(Book $book)
    {
        if(count($this->booksOnLoan) >= self::BOOK_LIMIT) {
            throw new TooManyBooksOnLoanException();
        }
        $this->booksOnLoan[] = $book;
    }
```

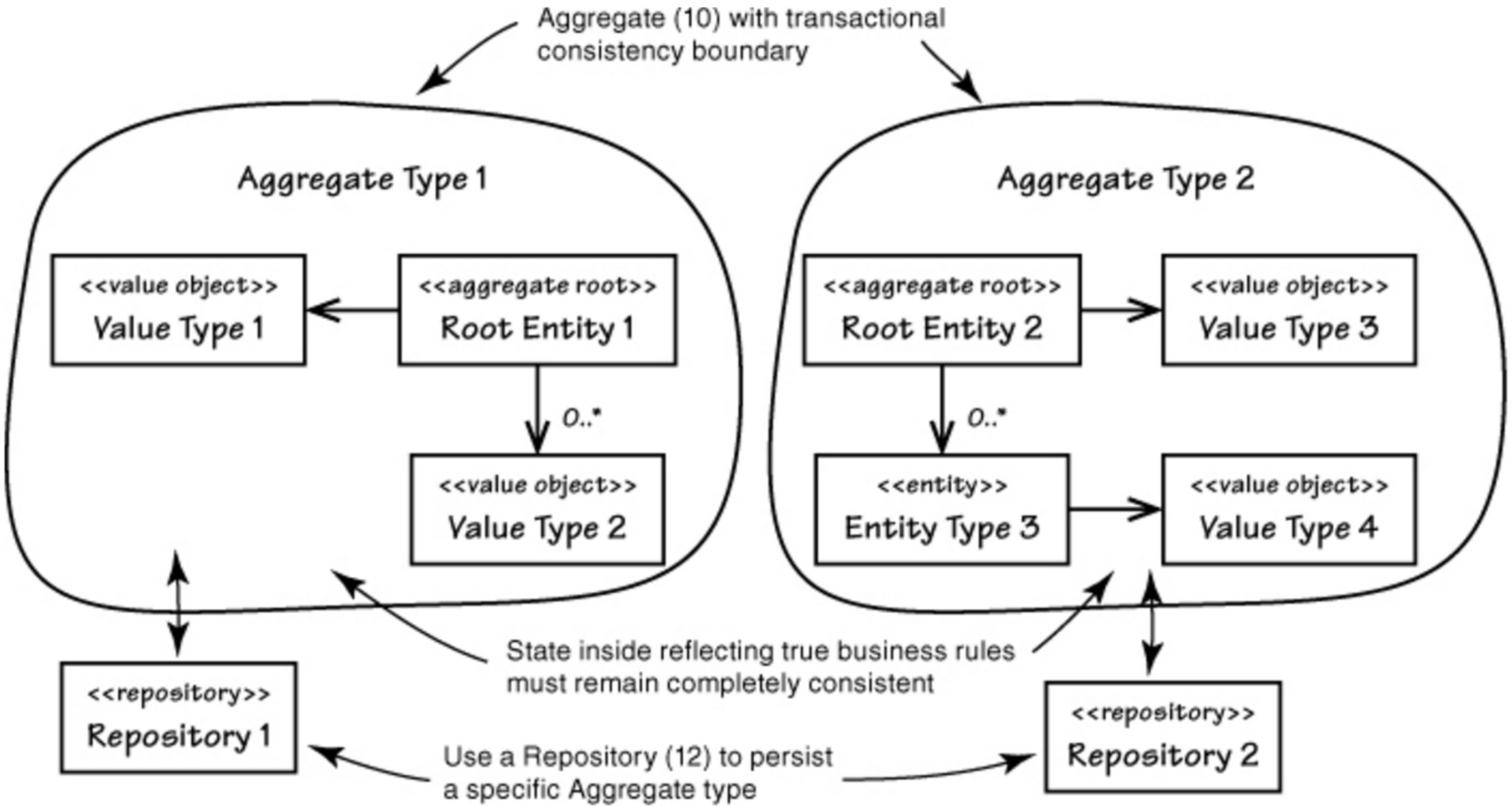


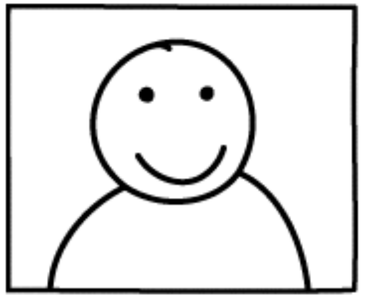
Figure G.4. Two Aggregate types with their own transactional consistency boundaries

Task Based UIs

A Web Page

http://www.wadenalibraryissuperreal.org/user/02294324-a7ef-4dfb-b5f3-590f3df10165

Wadena Library



Wade Wilson

53 2 St NE
Wadena, Saskatchewan S0A 4J0

Edit

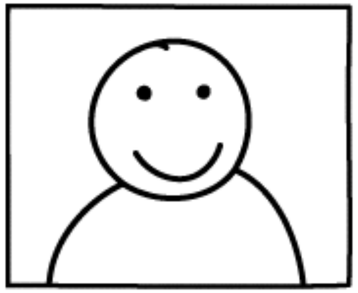
Borrowed Books (3) Add

Book Title	ISBN	Date Due	
Goodnight Unicorn: A Magical Parody	978-1934649633	04/04/2009	Edit Delete
How to Avoid Huge Ships	978-0870334337	04/01/2010	Edit Delete
Wolverine: Children Book of Fun Facts	978-1532799730	11/11/2015	Edit Delete

A Web Page

http://www.wadenalibraryissuperreal.org/user/02294324-a7ef-4dfb-b5f3-590f3df10165/wade-wilson

Wadena Library



Take New Photo

Wade Wilson

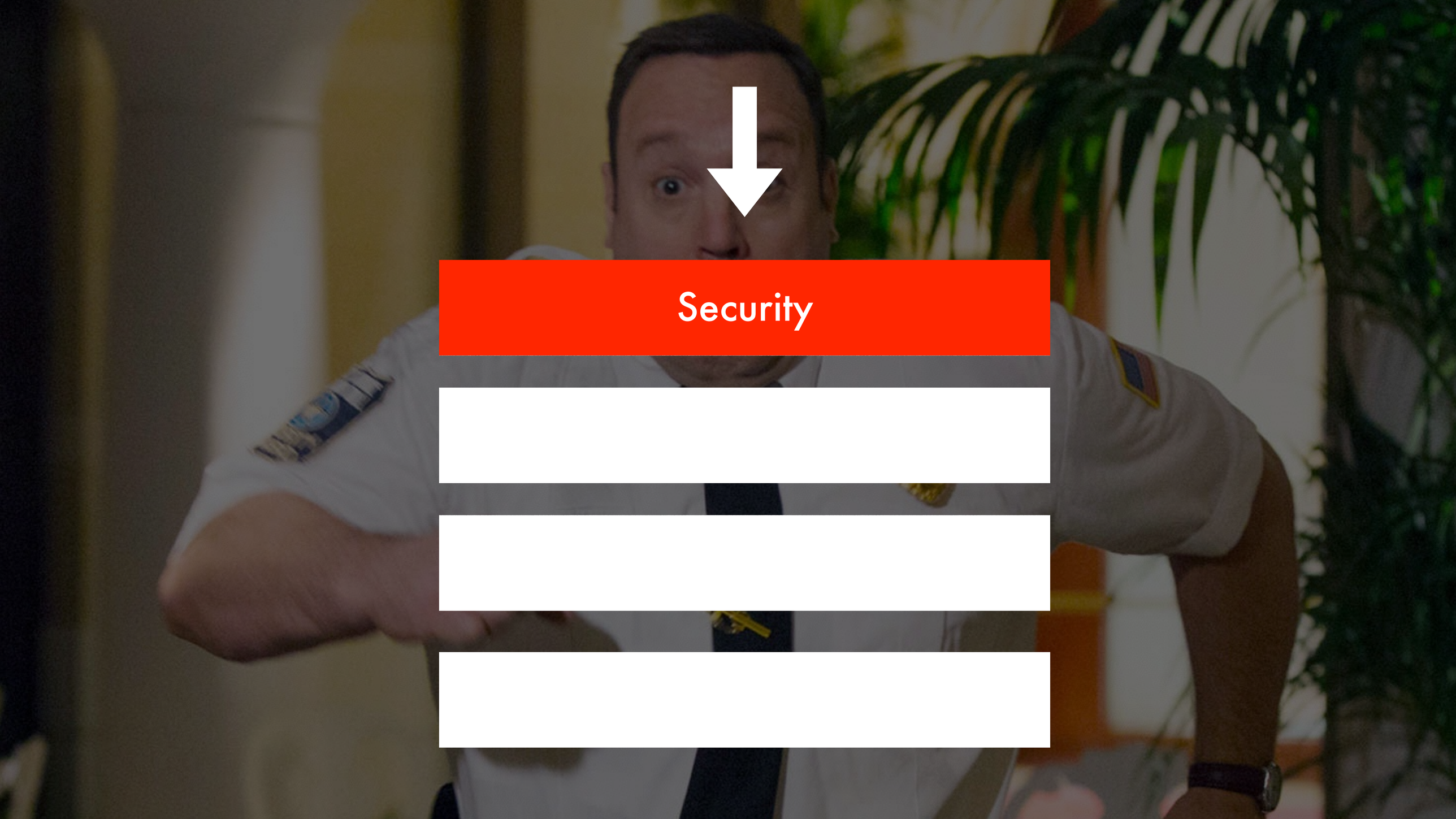
53 2 St NE
Wadena, Saskatchewan S0A 4J0

Change Address | Change Name

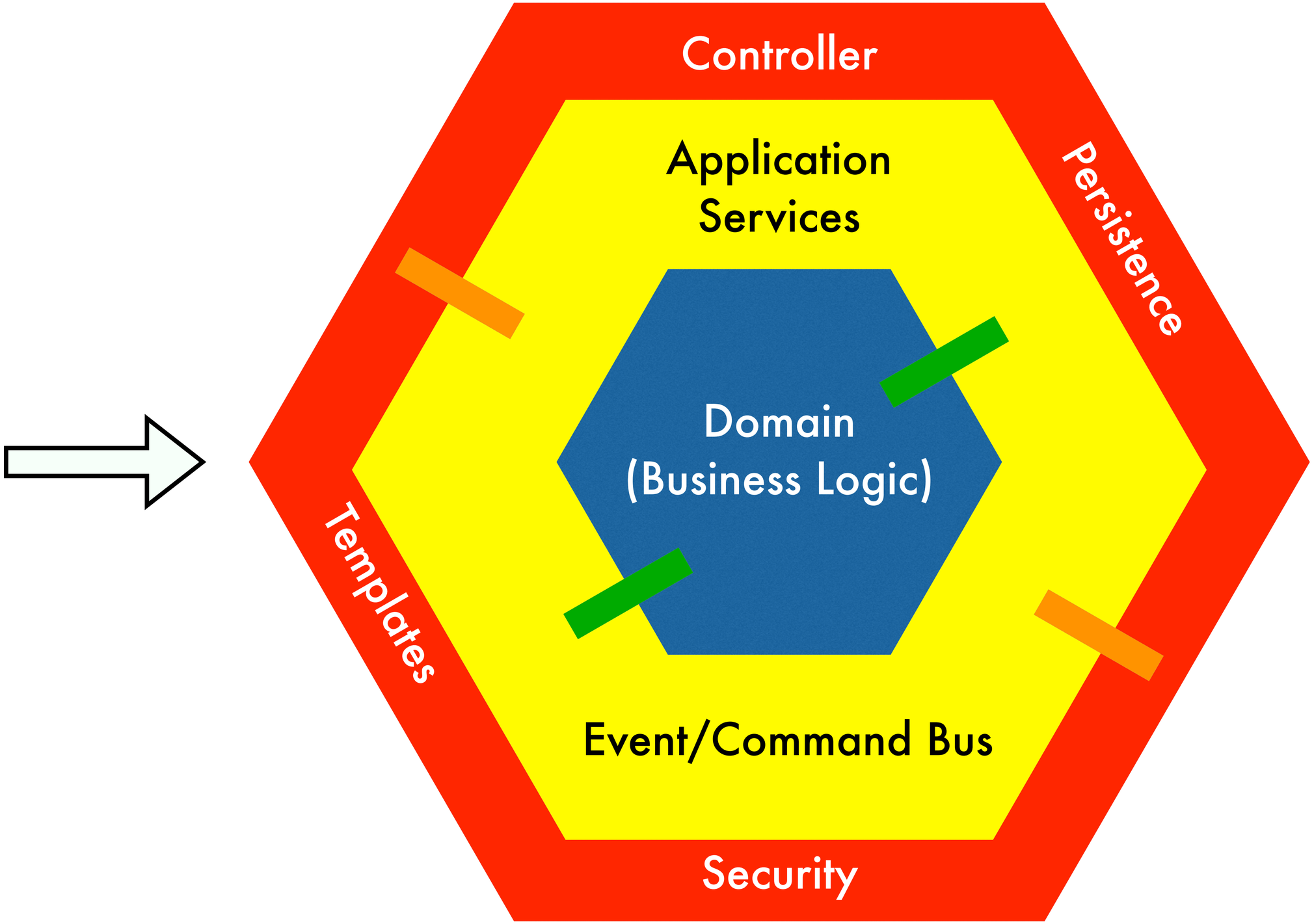
Borrowed Books (3) Borrow a Book

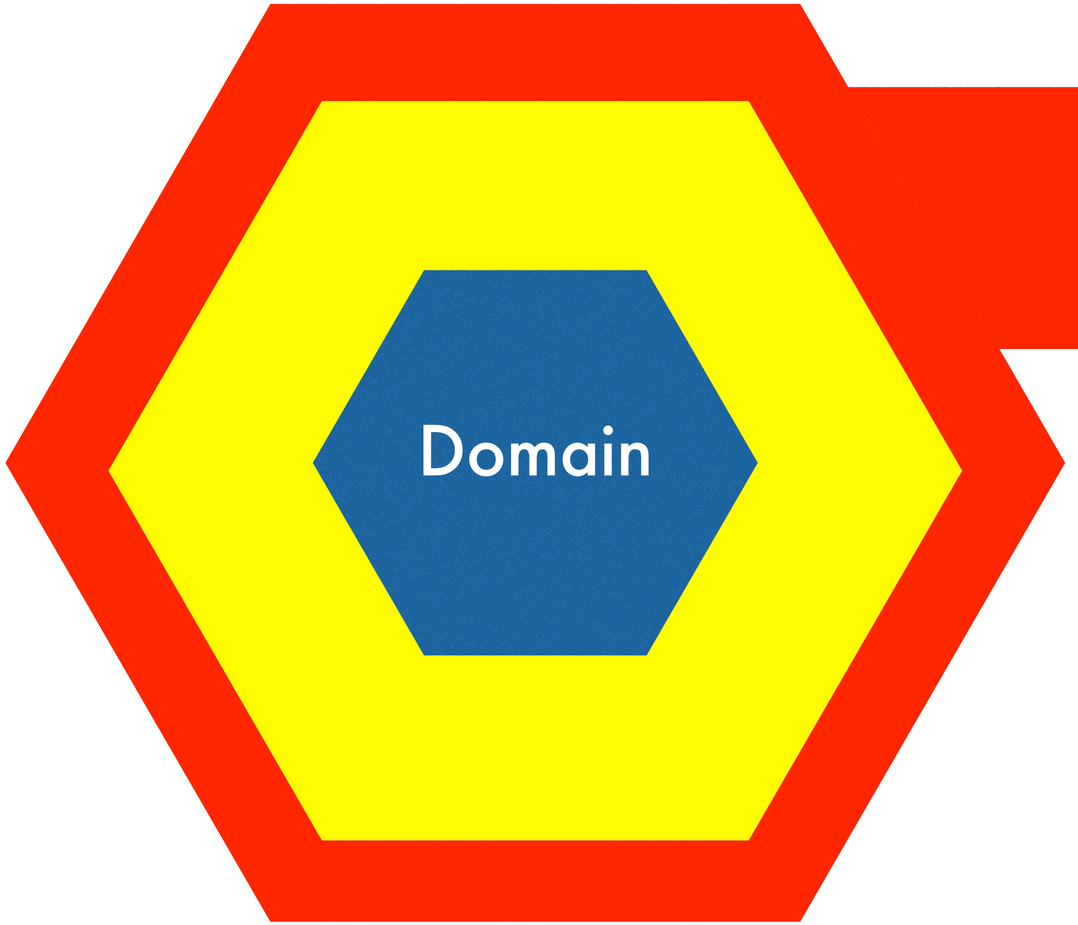
Book Title	ISBN	Date Due	
Goodnight Unicorn: A Magical Parody	978-1934649633	04/04/2009	Return Renew Replacement
How to Avoid Huge Ships	978-0870334337	04/01/2010	Return Renew Replacement
Wolverine: Children Book of Fun Facts	978-1532799730	11/11/2015	Return Renew Replacement

	DDD	BALL OF MUD/CRUD
Correctness	✓	
Testability	✓	
Usability	✓	
Maintainability		
Modifiability		

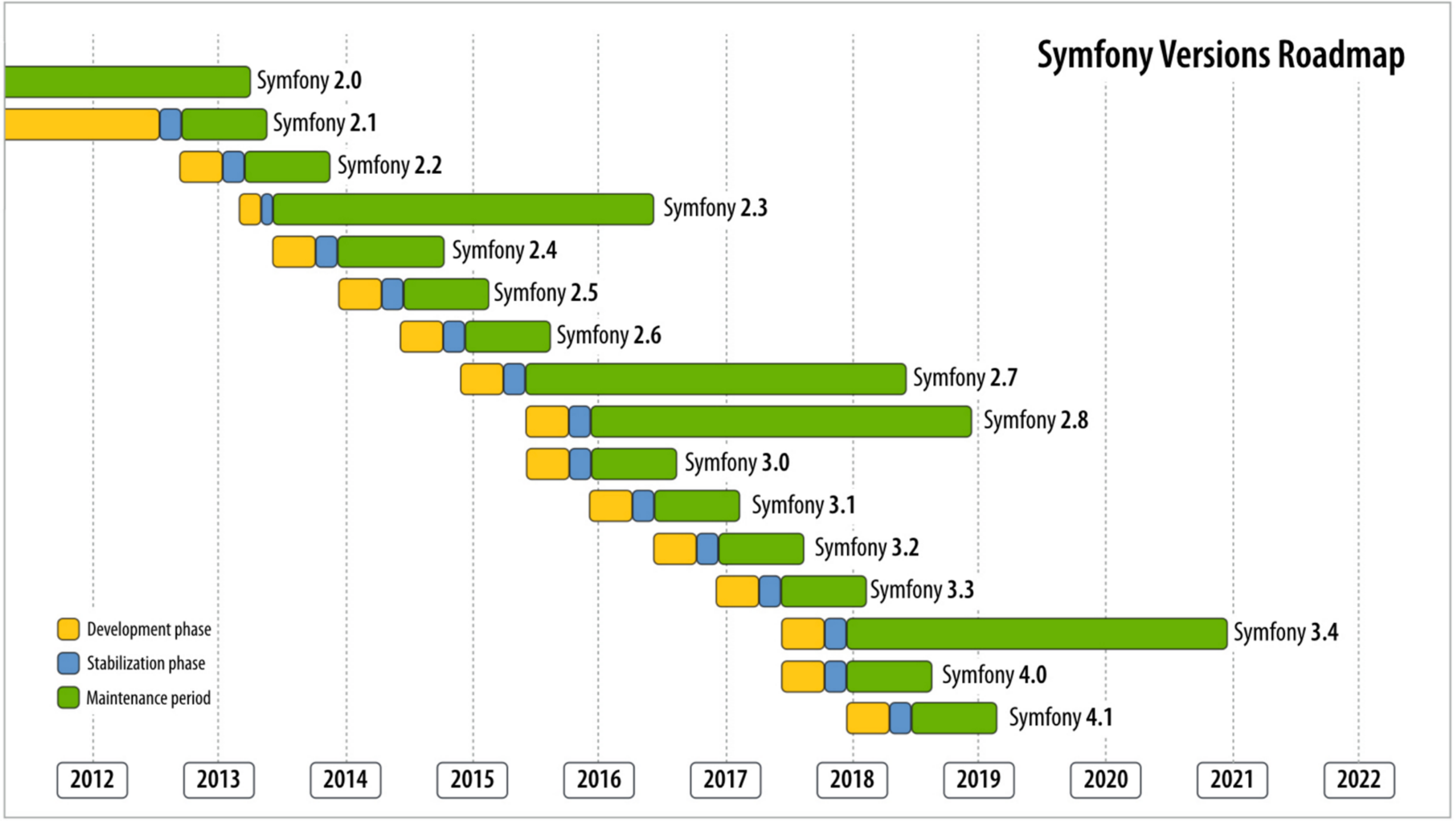


Security


Controller
Application
Services
Persistence
Domain
(Business Logic)
Templates
Event/Command Bus
Security



Framework


Symfony Versions Roadmap
Symfony 2.0
Symfony 2.1
Symfony 2.2
Symfony 2.3
Symfony 2.4
Symfony 2.5
Symfony 2.6
Symfony 2.7
Symfony 2.8
Symfony 3.0
Symfony 3.1
Symfony 3.2
Symfony 3.3
Symfony 3.4
Symfony 4.0
Symfony 4.1
Development phase
Stabilization phase
Maintenance period
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022

Proposed Release Schedule

June 2015

5.1 LTS

Bug fixes until June 2017, and security fixes until June 2018.

December 2015

Laravel 5.2

General release of 5.2 which includes 6 months of bug fixes, 1 year of security.

June 2016

Laravel 5.3

6 months of bug fixes, 1 year of security.

December 2016

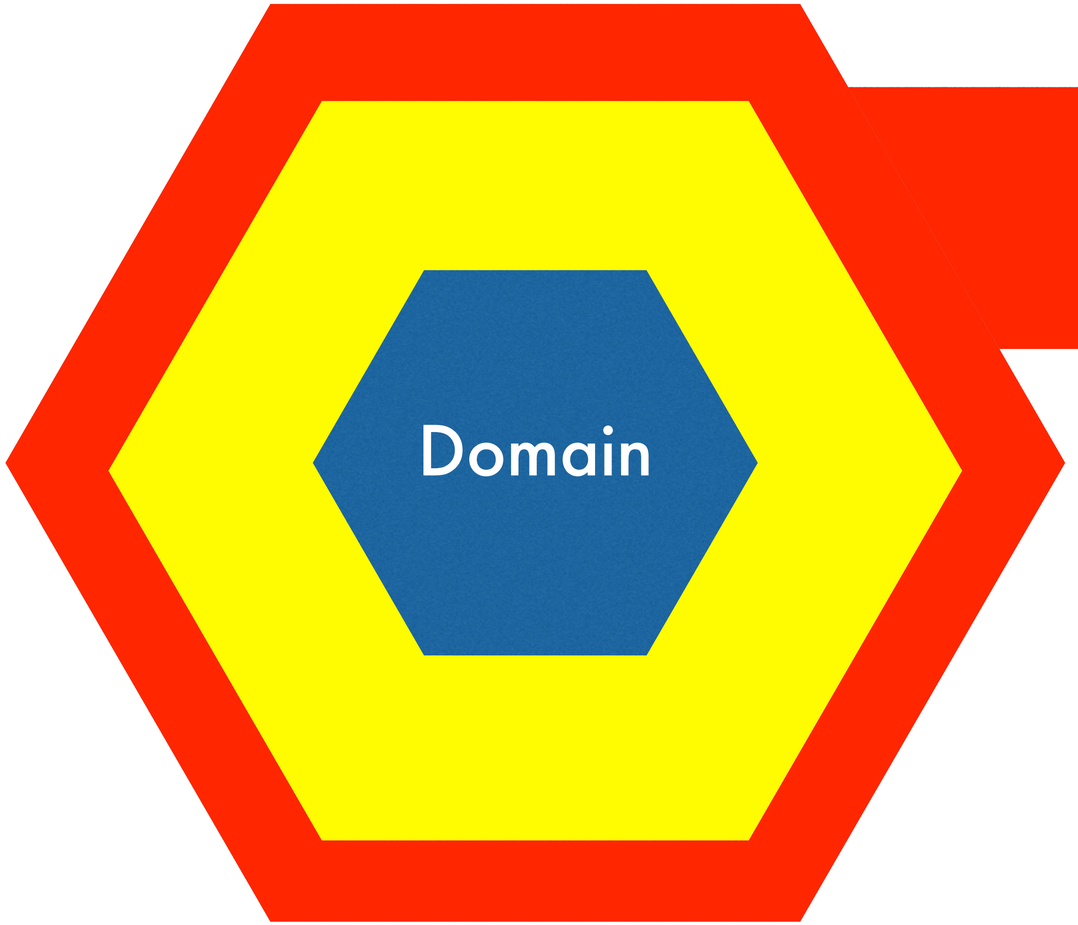
Laravel 5.4

6 months of bug fixes, 1 year of security.

June 2017

Laravel 5.5

The next LTS release. At this point Laravel 5.1 will no longer receive bug fixes, but will continue getting security fixes until June 2018.



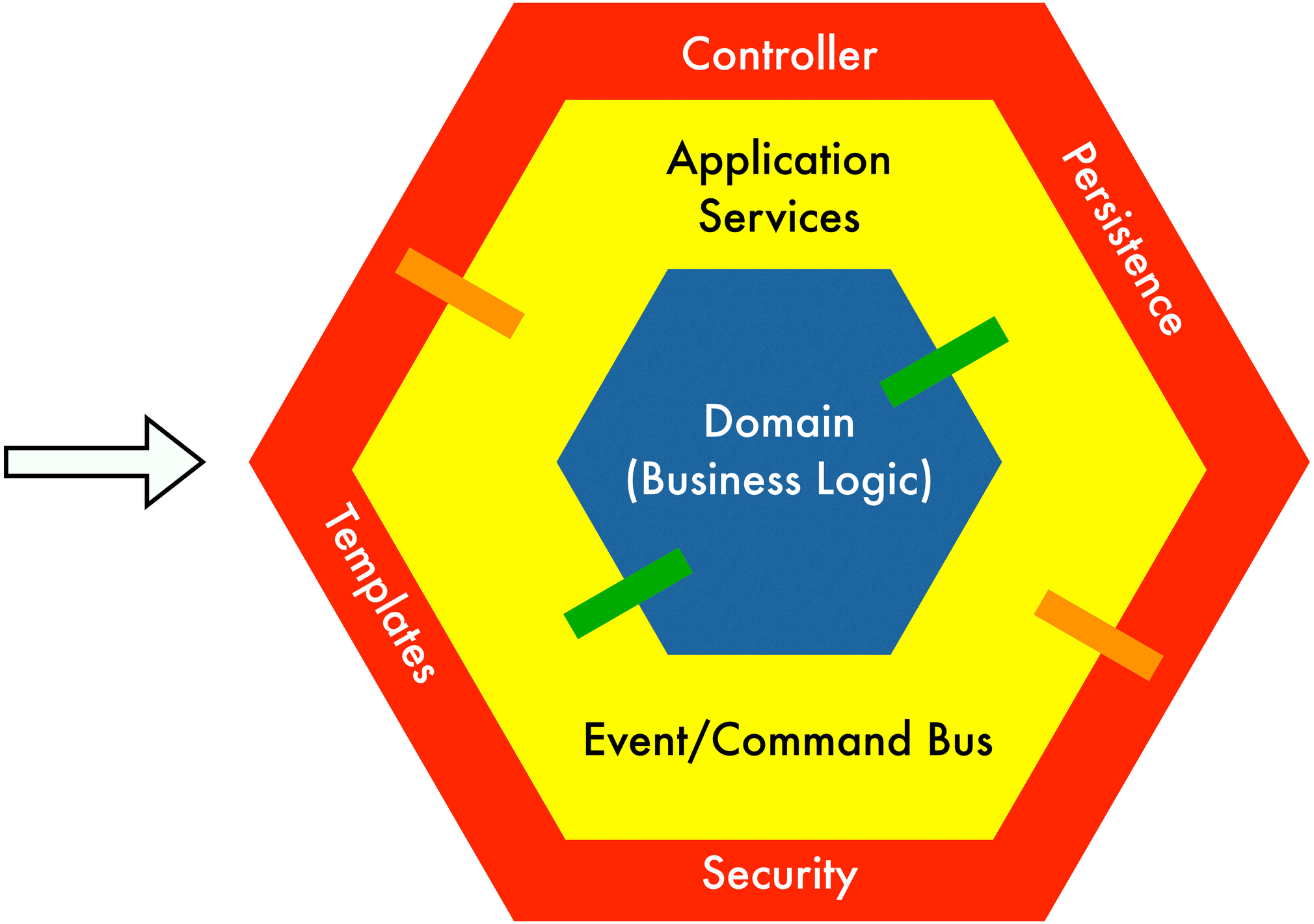
Framework

Mathias Verraes - Decoupling the Model from the Framework at Laracon EU 2014

https://www.youtube.com/watch?v=QaIGN_cTcc8

Ruby Midwest 2011 - Keynote: Architecture the Lost Years by Robert Martin

https://www.youtube.com/watch?v=WpkDN78P884


Controller
Application
Services
Persistence
Domain
(Business Logic)
Templates
Event/Command Bus
Security

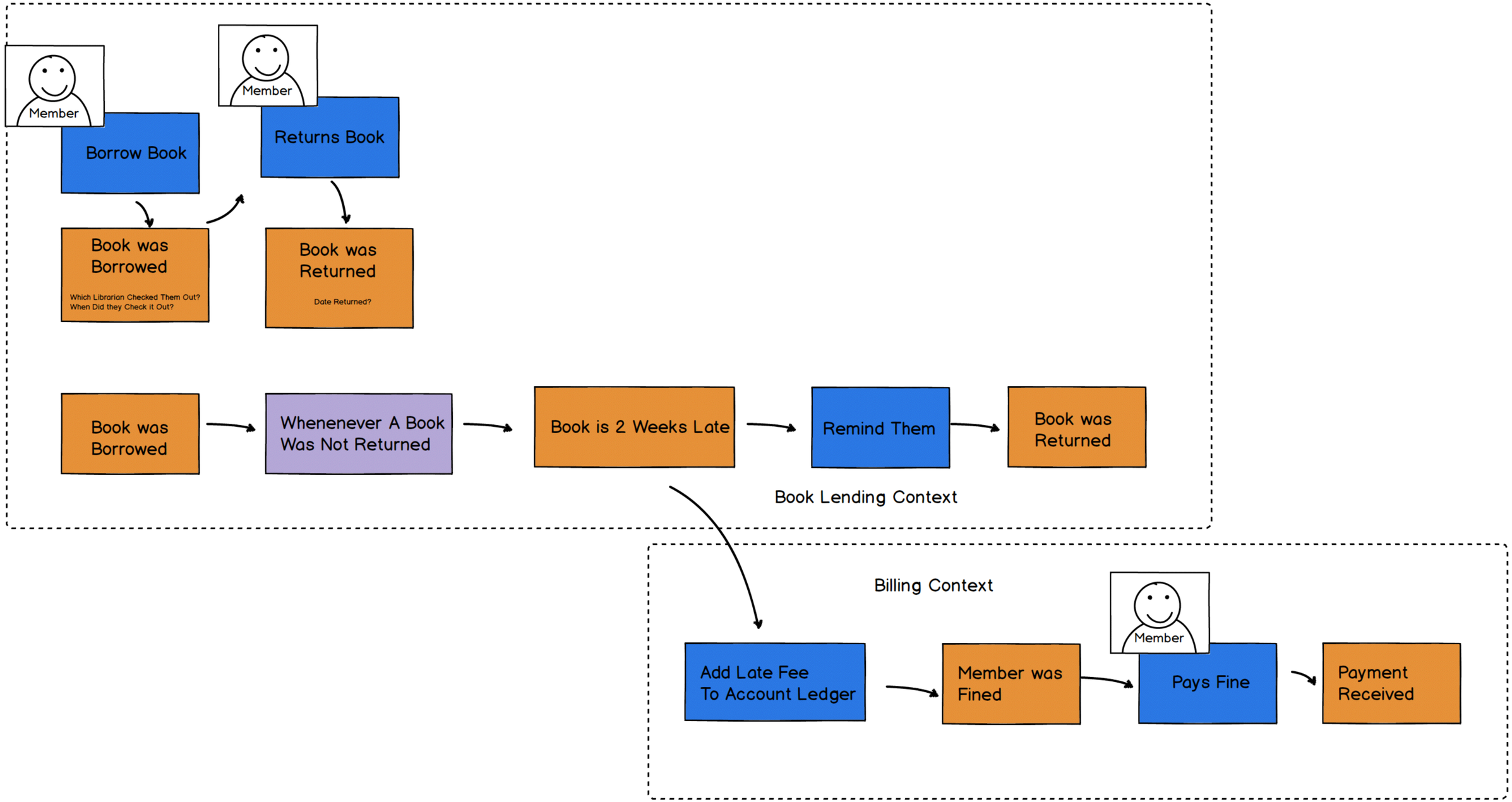
	DDD	BALL OF MUD/CRUD
Correctness	✓	
Testability	✓	
Usability	✓	
Maintainability	✓	
Modifiability		

Domain Events

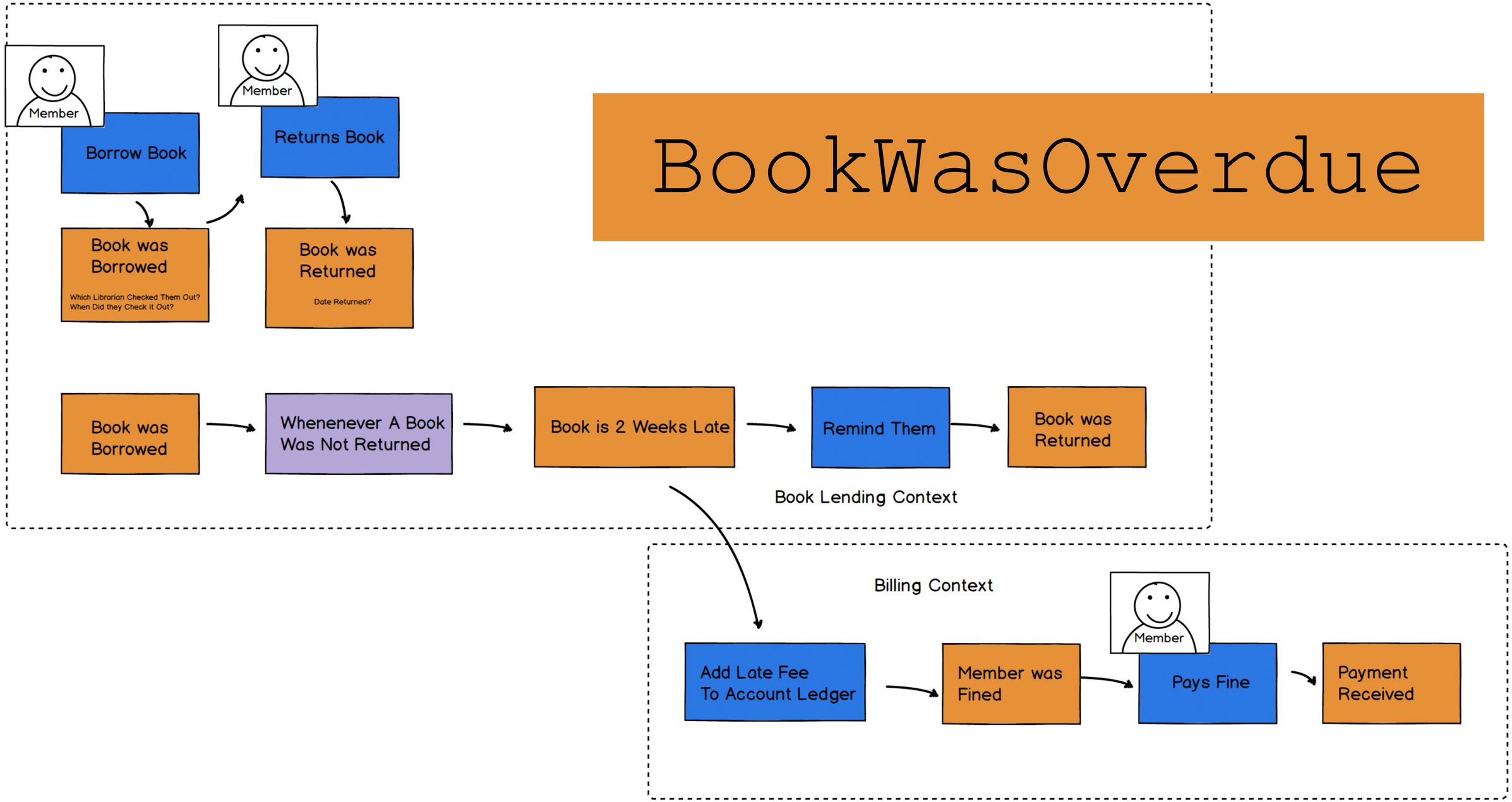
Domain Events

- Something That Happened In The Past
- Immutable
- Part of the Core Domain
- Usually Created in Either an Aggregate or as Part of Application Service
- Probably Want To Persist Them
- Passed to Event Bus (Application Service)
- Listeners are Application Services
- Used To Cross Bounded Contexts

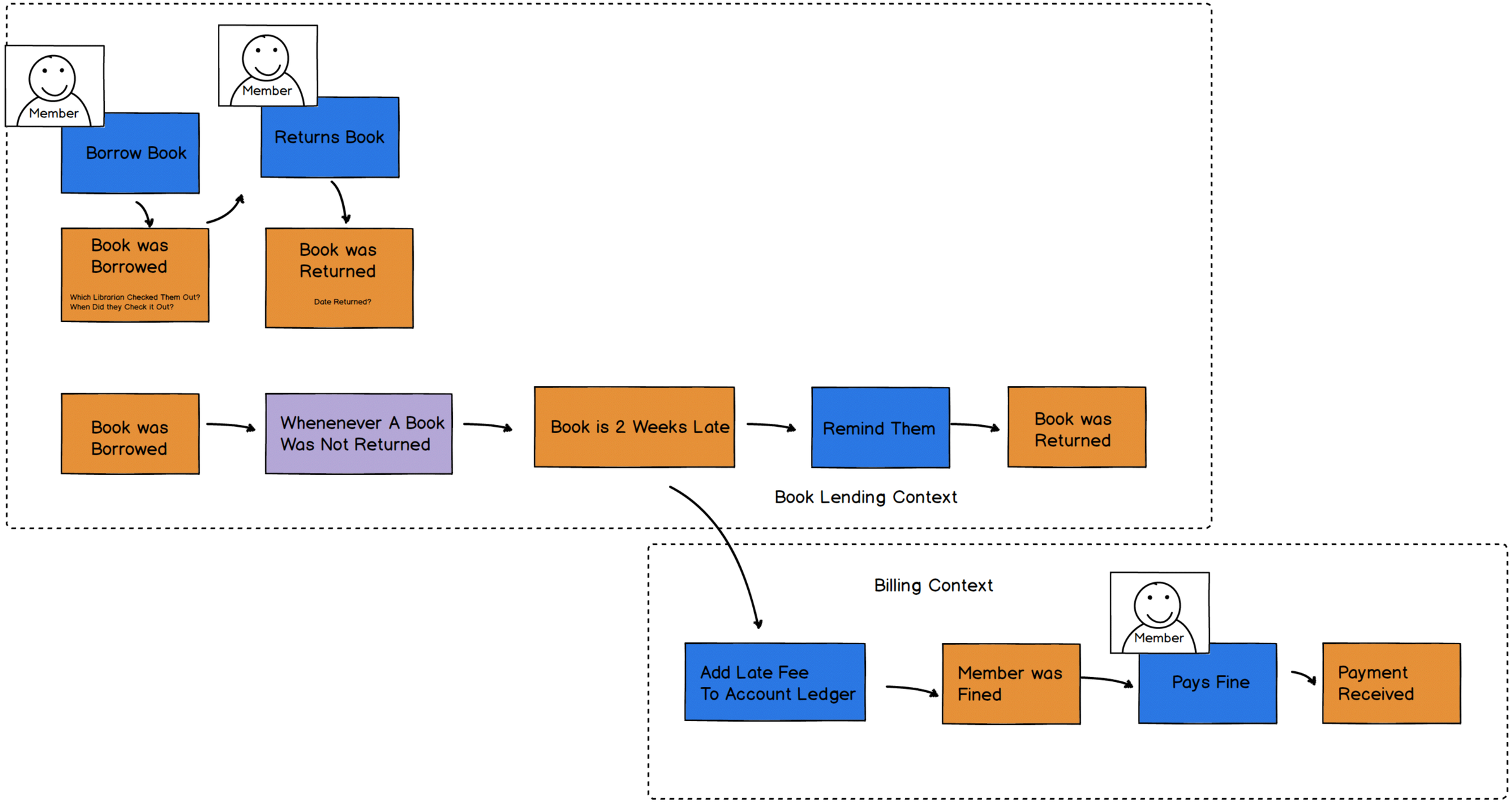
```php
class BookWasBorrowed
{
    /**
     * @var BookId
     */
    private $bookId;
    /**
     * @var DateBorrowed
     */
    private $dateBorrowed;
    /**
     * @var MemberId
     */
    private $memberId;

    public function __construct(DateBorrowed $dateBorrowed, BookId $bookId, MemberId $memberId)
    {
        $this->bookId = $bookId;
        $this->dateBorrowed = $dateBorrowed;
        $this->memberId = $memberId;
    }

}
```


Member
Borrow Book
Member
Returns Book
Book was Borrowed
Which Librarian Checked Them Out?
When Did they Check it Out?
Book was Returned
Date Returned?
Book was Borrowed
Whenenever A Book Was Not Returned
Book is 2 Weeks Late
Remind Them
Book was Returned
Book Lending Context
Billing Context
Member
Add Late Fee To Account Ledger
Member was Fined
Pays Fine
Payment Received

BookWasOverdue



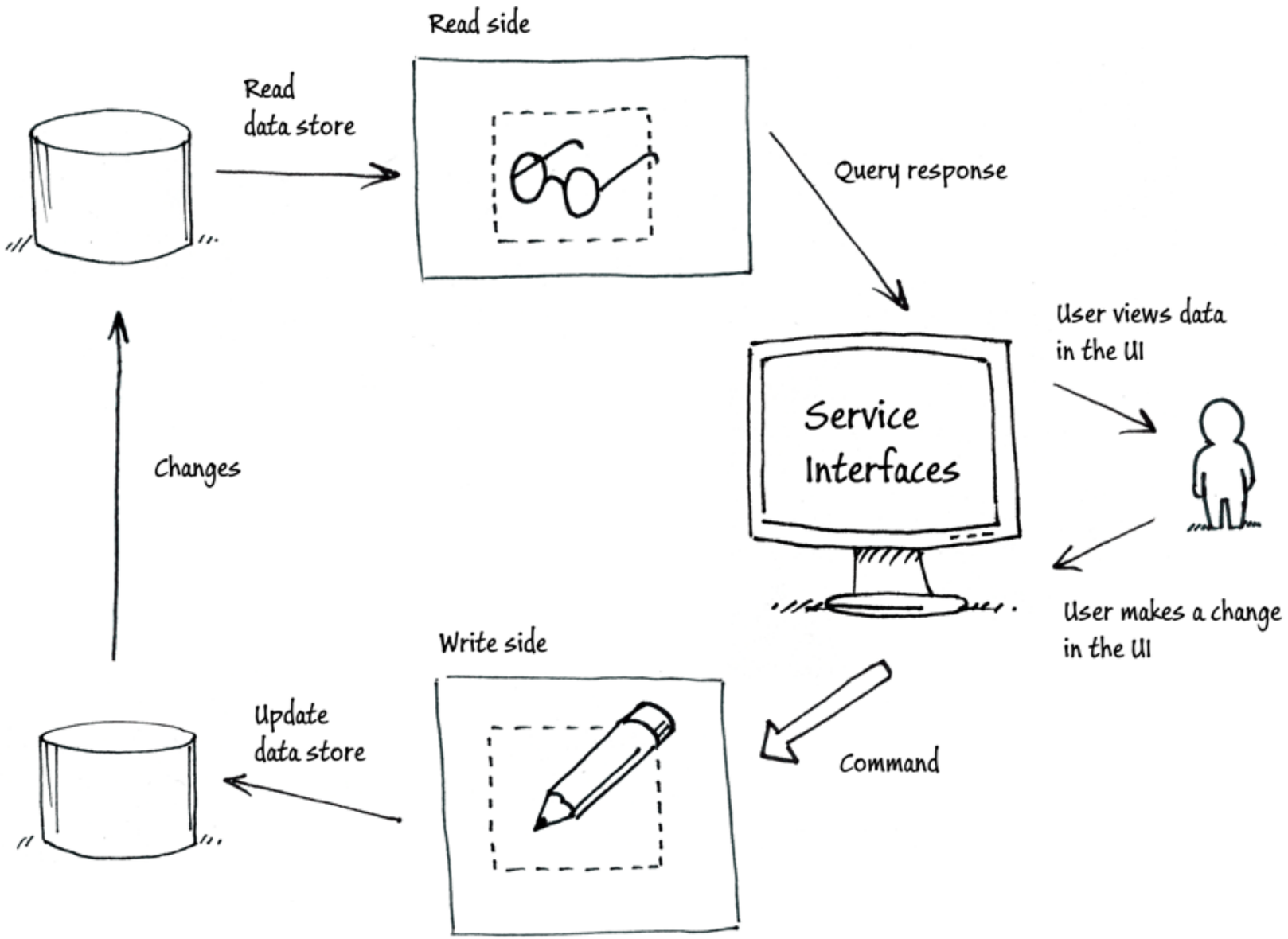
```php
class AccountLedgerListener
{
    // ...

    public function onBookWasOverDueAction(BookWasOverDue $event)
    {
        $fineAmount = new FineAmount(.10);
        if($event->getDateDue()->daysSince() > 14) {
            $member = $this->membersRepository->findById($event->getMemberId());
            $member->chargeFine($fineAmount);
            $this->membersRepository->updateAccount($member);

            $this->eventBus->publish(new MemberWasFined($member->getId(),$fineAmount));
        }
    }
```


Member
Borrow Book
Member
Returns Book
Book was Borrowed
Which Librarian Checked Them Out?
When Did they Check it Out?
Book was Returned
Date Returned?
Book was Borrowed
Whenenever A Book Was Not Returned
Book is 2 Weeks Late
Remind Them
Book was Returned
Book Lending Context
Billing Context
Member
Add Late Fee To Account Ledger
Member was Fined
Pays Fine
Payment Received

	DDD	BALL OF MUD/CRUD
Correctness	✓	
Testability	✓	
Usability	✓	
Maintainability	✓	
Modifiability	✓	

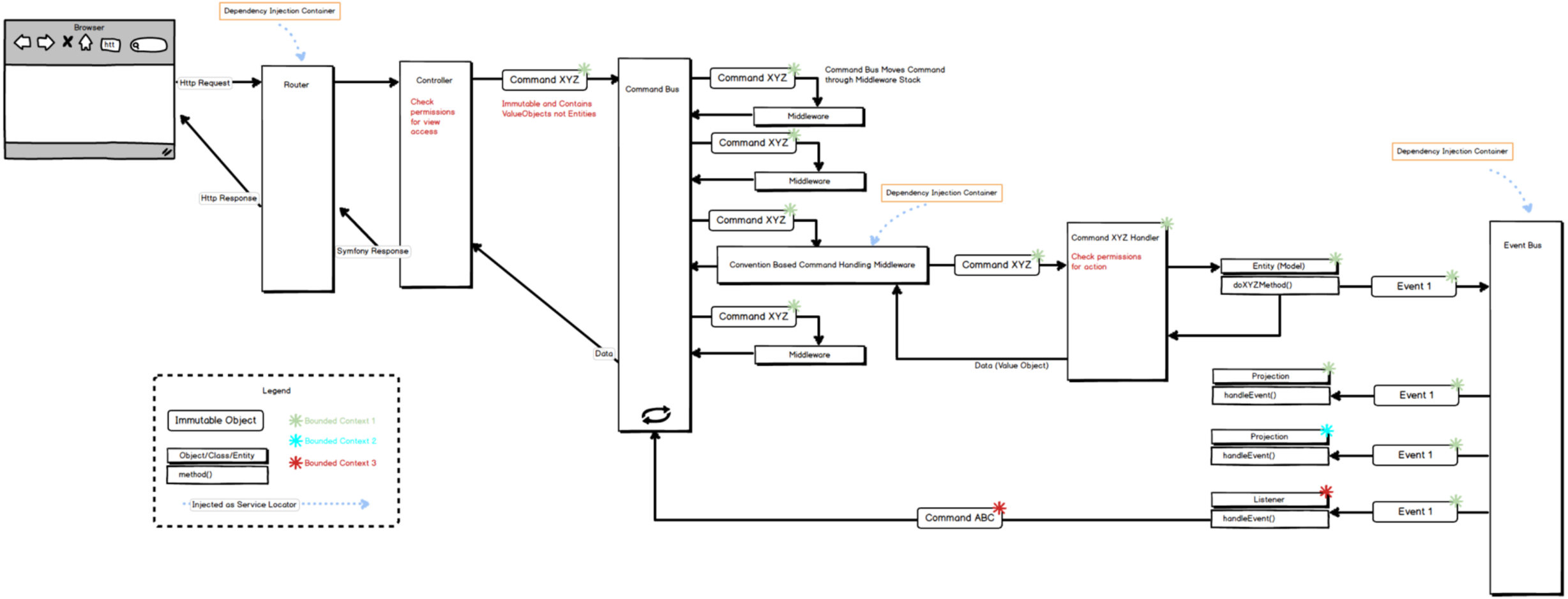
Command Query Responsibility Segregation

CQRS

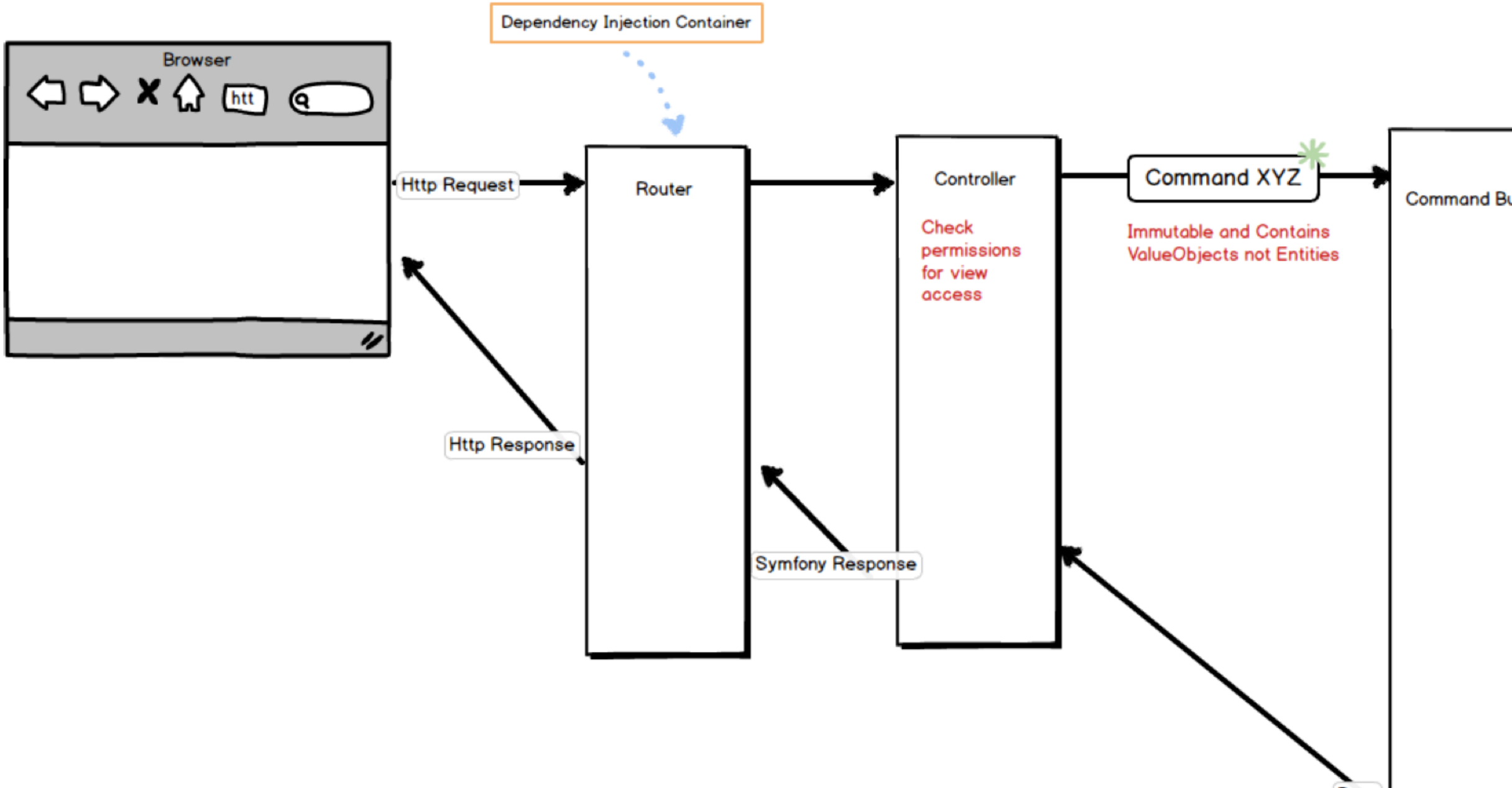


Slide: Microsoft

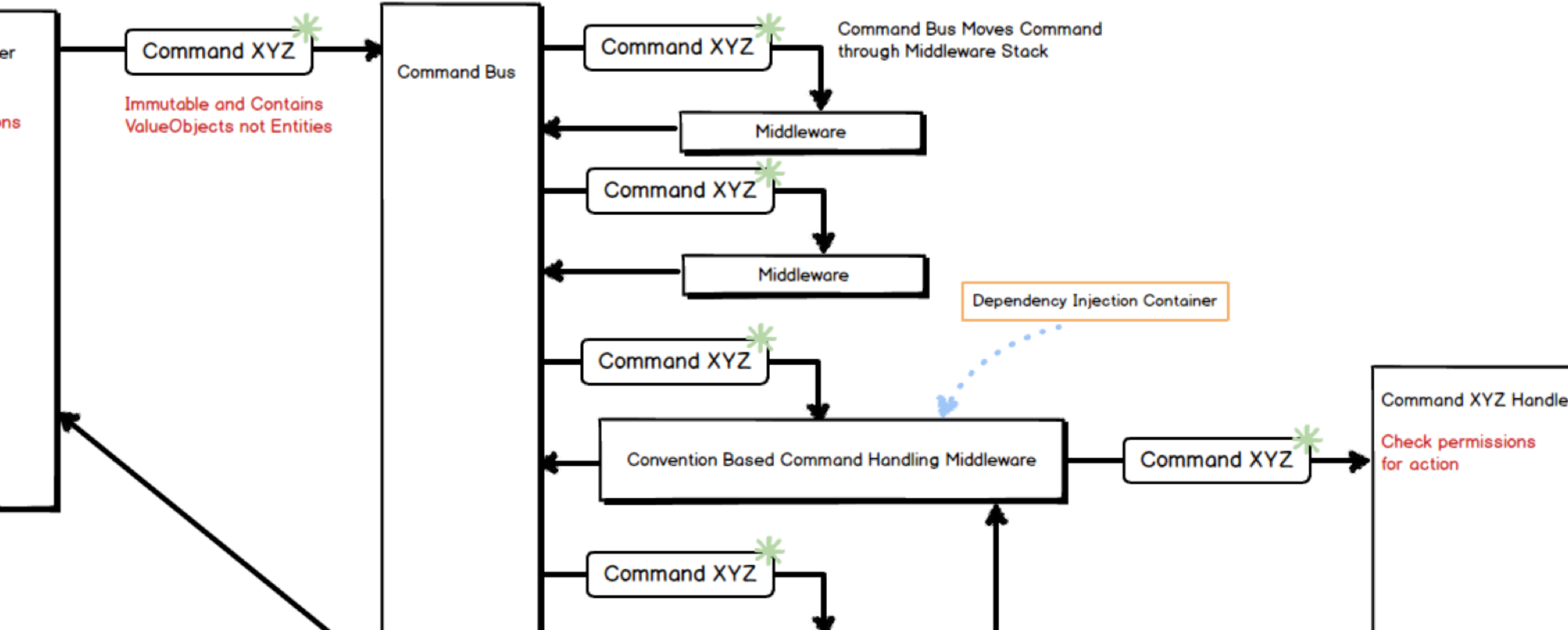
CQRS Command and Event Architecture

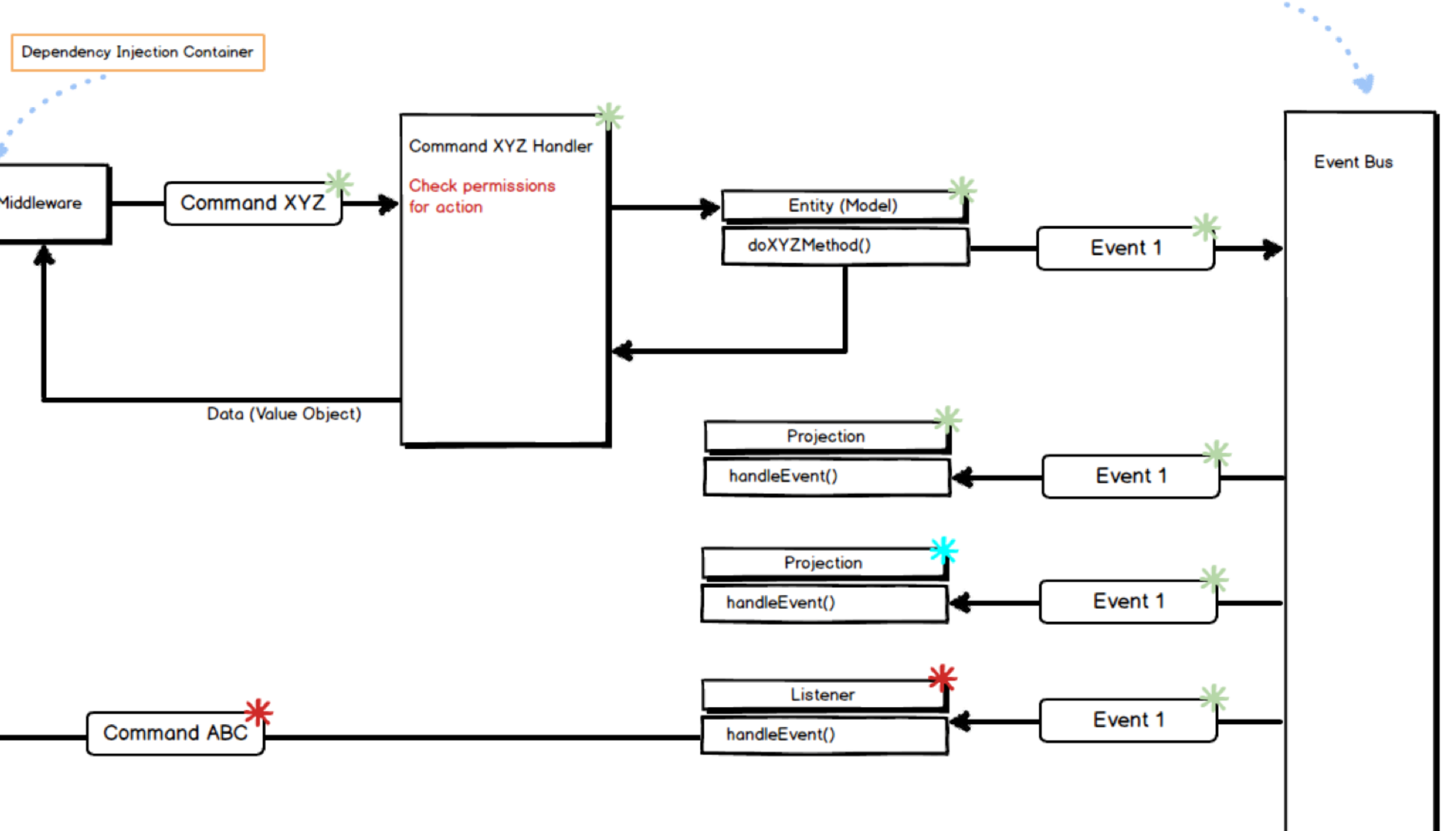


CQRS


Dependency Injection Container
Browser
htt
Http Request
Router
Controller
Check permissions for view access
Command XYZ
Immutable and Contains ValueObjects not Entities
Command Bu
Http Response
Symfony Response

CQRS Command and Event Architecture



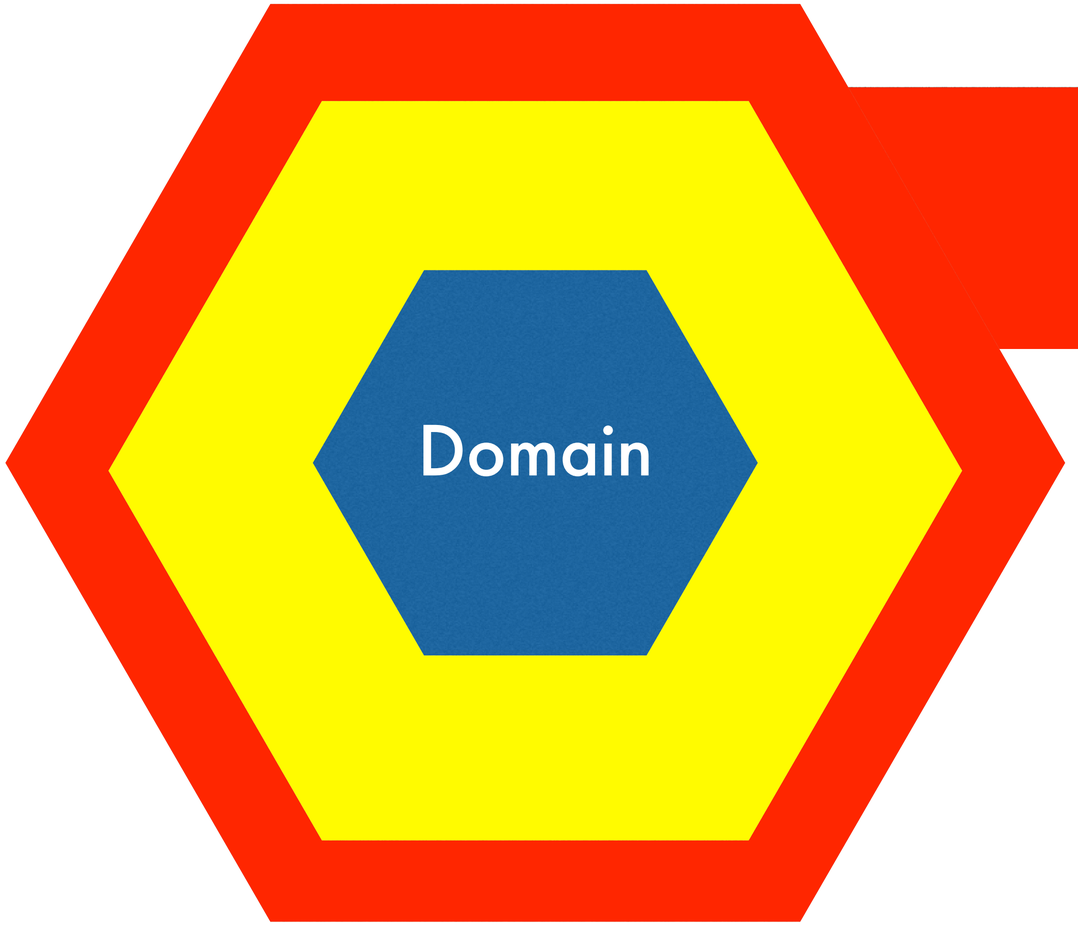

Dependency Injection Container
Middleware
Command XYZ
Command XYZ Handler
Check permissions for action
Entity (Model)
doXYZMethod()
Event 1
Event Bus
Data (Value Object)
Projection
handleEvent()
Event 1
Projection
handleEvent()
Event 1
Listener
handleEvent()
Event 1
Command ABC

https://gnugat.github.io/2016/05/11/towards-cqrs-command-bus.html

```
class ProfileCreationController extends Controller
{
    /**
     * @Route("/api/v1/profiles")
     * @Method({"POST"})
     */
    public function createProfileAction(Request $request)
    {
        $em = $this->get('doctrine.orm.entity_manager');

        $name = $request->request->get('name');
        if (null === $name) {
            return new JsonResponse(array('error' => 'The "name" parameter is missing from the r
        }
        if (null !== $em->getRepository('AppBundle:Profile')->findOneByName($name)) {
            return new JsonResponse(array('error' => 'The name "'.$name.'" is already taken'), 4
        }
        $createdProfile = new Profile($name);
        $em->persist($createdProfile);
        $em->flush();
```

```php
class ProfileCreationController extends Controller
{
    /**
     * @Route("/api/v1/profiles")
     * @Method({"POST"})
     */
    public function createProfileAction(Request $request)
    {
        try {
            $createdProfile = $this->get('command_bus')->handle(new CreateNewProfile(
                $request->request->get('name')
            ));
        } catch (\DomainException $e) {
            return new JsonResponse(array('error' => $e->getMessage()), 422);
        }

        return new JsonResponse($createdProfile->toArray(), 201);
    }
}
```

```php
class CreateNewProfileHandler
{
    private $checkProfileNameDuplicates;
    private $saveNewProfile;

    public function __construct(
        CheckProfileNameDuplicates $checkProfileNameDuplicates,
        SaveNewProfile $saveNewProfile
    ) {
        $this->checkProfileNameDuplicates = $checkProfileNameDuplicates;
        $this->saveNewProfile = $saveNewProfile;
    }

    public function handle(CreateNewProfile $createNewProfile)
    {
        if (true !== $this->checkProfileNameDuplicates->check($createNewProfile->name)) {
            throw new \DomainException("Invalid \"name\" parameter: \"$name\" already exists and
        }
        $newProfile = new Profile($name); // Entity moved to Profile namespace
        $this->saveNewProfile->save($newProfile);

        return $newProfile
    }
}
```



Framework

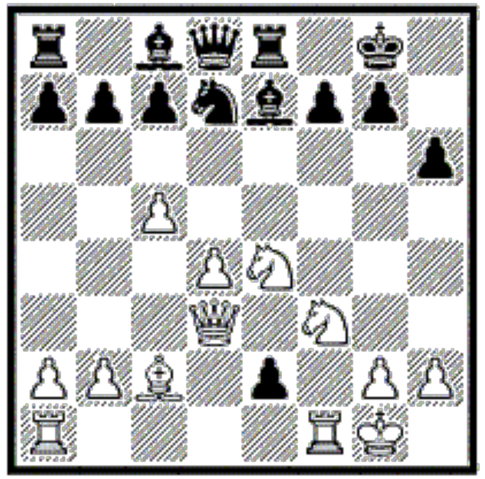
Event Sourcing

Event Sourcing

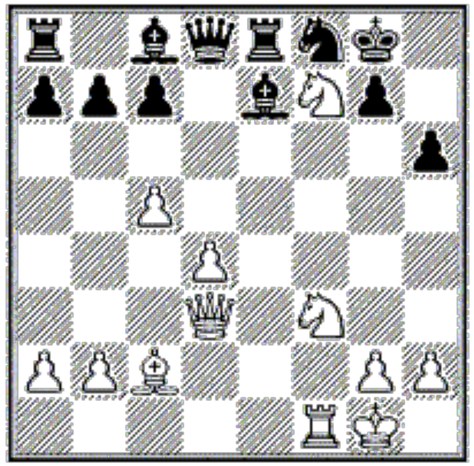
- Store the Events That Happen
- Append Only Event Storage
- Object Properties are Not Stored

Spassky- Bronstein
1960 USSR Championship
Annotated by Spassky

1. e4 e5 2. f4 ef4 3. ♘f3 d5 4. ed5 ♗d6 5. ♘c3 ♘e7 6. d4 O-O 7. ♗d3 ♘d7 8. O-O h6? [8. ... ♘g6 9. ♘e4 ♘f6 10. ♘d6 ♕d6 11. c4 ♗g4=; 8. ... ♘f6 9. ♘e5 ♘ed5 10. ♘d5 ♘d5 11 ♗f4 ♘f4 12. ♖f4 ♕g5=] **9. ♘e4 ♘d5 10. c4 ♘e3 11. ♗e3 fe3 12. c5 ♗e7** [12. ... ♗f4? 13. g3 ♗g5 14. ♘fg5 hg5 15. 15. ♕h5±; 13. ... f5 14. ♘c3 ♗g5 15. h4 ♗e7 16. ♘d5±] **13. ♗c2! ♖e8** [13. ... ♘f6 14 ♕d3 ♘e4 15. ♕e4 g6 16. ♕e3 ♔g7± → »] **14. ♕d3 e2**



15. ♘d6!? [15. ♖f2!±] **♘f8?** [15. ... ef1=♕ 16. ♖f1 ♘f6 17. ♘f7 ♔f7 18. ♘e5 ♔g8 ♕h7! ♘h7 20. ♗b3+−; ⌓15. ... ♗d6 16. ♕h7 ♔f8 17. cd6 ef1=♕ 18. ♖f1 cd6 19. ♕h8 ♔e7 20. ♖e1 ♘e5 21. ♕g7 ♖g8 22. ♕h6 ♕b6 23. ♔h1 ♗e6 24. de5±] **16. ♘f7! ef1=♕ 17. ♖f1**



17. ...♗f5 [17. ... ♔f7 18. ♘e5 ♔g8 19. ♕h7 ♘h7 20. ♗b3+−; 17. ... ♕d5 18. ♗b3 ♕f7 19. ♗f7 ♔f7 20. ♕c4 ♔g6 21. ♕g8 ♗f6 22. ♘h4 ♗h4 23. ♕f7 ♔h7 24. ♕e8+−; 22. ... ♔g5 23. ♕d5 ♔h4 24. ♖f4 ♗g4 25. g3 ♔h3 26. ♕g2#] **18. ♕f5 ♕d7 19. ♕f4** [19. ♕d3!+−] **♗f6 20. ♘3e5 ♕e7 21. ♗b3 ♗e5 22. ♘e5 ♔h7 23. ♕e4!** [△♖f8+−] 1-0

Rock Castle Construction

Transaction Journal

All Transactions

	Item	Item Description	Account	Class	Sales Price	Amount
			11000 · Accounts Receivable	Remodel		5,019.08
or	Removal	Removal labor	40130 · Labor Income	Remodel	35.00	-560.00
...	Subs:Ele...	Electrical work	40150 · Subcontracted Labor Inc...	Remodel	57.75	-231.00
	Subs:Plu...	Plumbing	40150 · Subcontracted Labor Inc...	Remodel	52.50	-420.00
...	Subs:Til...	Install tile or counter	40150 · Subcontracted Labor Inc...	Remodel	42.00	-504.00
	Subs:Pa...	Painting	40150 · Subcontracted Labor Inc...	Remodel	36.75	-147.00
	Appliance		40140 · Materials Income	Remodel	430.00	-430.00
	Appliance		40140 · Materials Income	Remodel	600.00	-600.00
...	Counter	Custom made coun...	40140 · Materials Income	Remodel	1,900.00	-1,900.00
...	San Tom...	CA sales tax, San...	25500 · Sales Tax Payable	Remodel	7.75%	-227.08
						0.00
						0.00

symfony / symfony

Watch 1,086 | Star 12,407 | Fork 4,850

Code | Issues 599 | Pull requests 171 | Pulse | Graphs

Catch \Throwable #18765

Merged **fabpot** merged 1 commit into `symfony:2.8` from `fprochazka:php7-throwable` 3 days ago

Conversation 103 | Commits 1 | Files changed 1

Showing changes from **all commits** | **1 changed file** | +1 −1 | Diff options

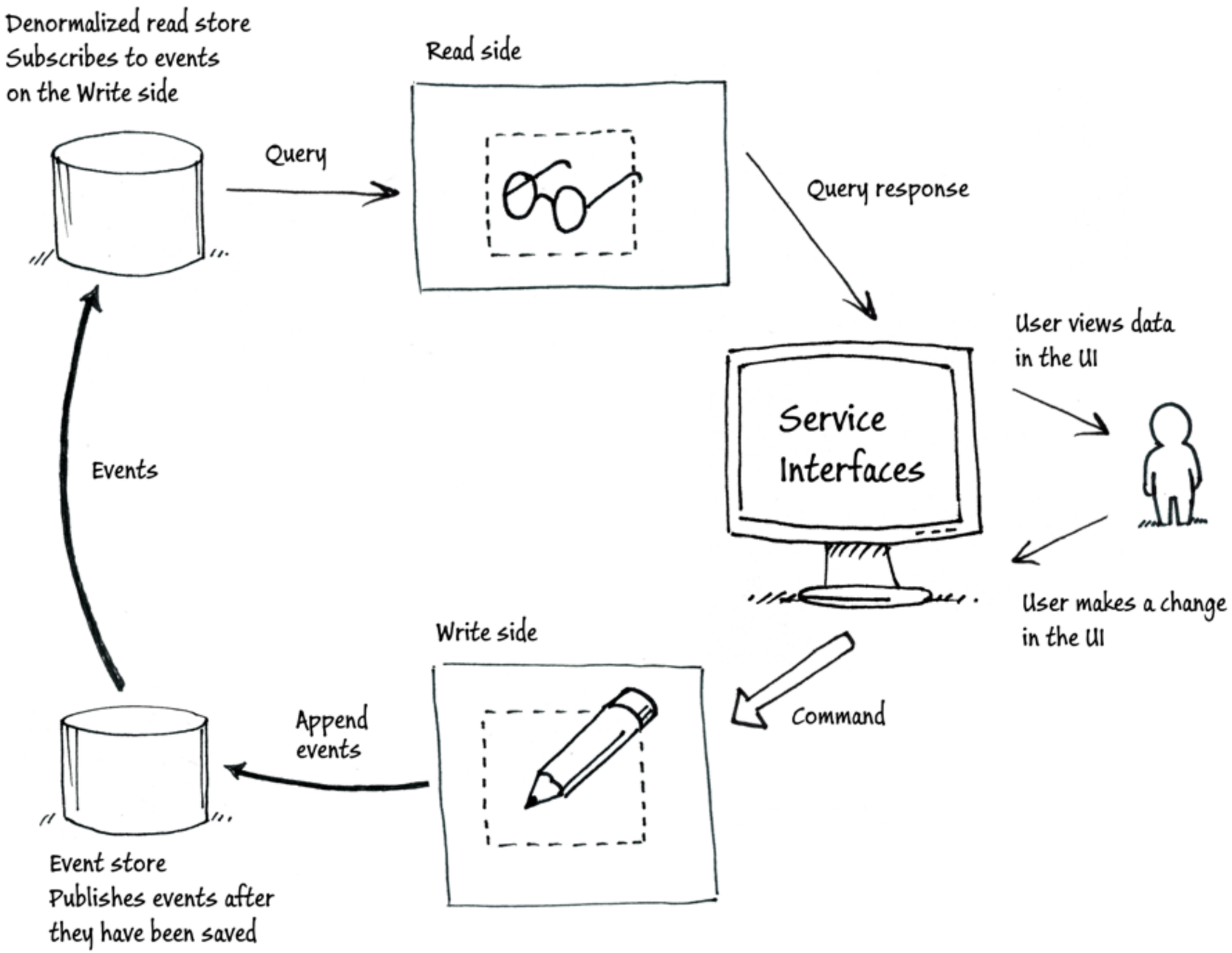
2 src/Symfony/Component/DependencyInjection/Compiler/AutowirePass.php — View

```
@@ -44,8 +44,8 @@ public function process(ContainerBuilder $container)
44                     $this->completeDefinition($id, $definition);
45                 }
46             }
47 -        } catch (\Error $e) {
48         } catch (\Exception $e) {
49         }
50
51         spl_autoload_unregister($throwingAutoloader);
```

```
@@ -44,8 +44,8 @@ public function process(ContainerBuilder $container)
44                     $this->completeDefinition($id, $definition);
45                 }
46             }
47         } catch (\Exception $e) {
48 +       } catch (\Throwable $e) {
49         }
50
51         spl_autoload_unregister($throwingAutoloader);
```

Event Sourcing

- Avoids Data Mapping
- Avoids Object-relational Impedance Mismatch
- Reduces Database Table Counts (Related Tables)
- Potentially Reduces Model Counts



Slide: Microsoft

```php
class Member
{
    /**
     * Member constructor.
     * @param MemberHistory $memberHistory
     */
    public function __construct(MemberHistory $memberHistory)
    {
        $this->memberHistory = $memberHistory;
        $this->memberId = new MemberId(); //uuid

        foreach ($memberHistory as $event) {
            $this->callEventApplyMethod($event); // also called on Doctrine postLoad
        }
    }
```

```php
class Member
{
    /**
     * Member constructor.
     * @param MemberHistory $memberHistory
     */
    public function __construct(MemberHistory $memberHistory)
    {
        $this->memberHistory = $memberHistory;
        $this->memberId = new MemberId(); //uuid

        foreach ($memberHistory as $event) {
            $this->applyEvent($event); // also called on Doctrine postLoad
        }
    }
```

```php
private function applyEvent(MemberEvent $event)
{
    if (! $event->getMemberId()->equals($this->getMemberId())) {
        throw new \Exception("Unable to apply event. Event does apply to this record.");
    }

    $method = $this->getApplyMethod($event);

    if (! method_exists($this, $method)) {
        throw new \Exception("Unable to apply event. Method:" . $method . " does not exist");
    }

    $this->$method($event);

}

private function getApplyMethod(MemberEvent $event)
{
    $classParts = explode('\\', get_class($event));

    return 'apply' . end($classParts) . 'Event';
}
```

```php
    private function applyBookWasBorrowedEvent(BookWasBorrowed $bookWasBorrowed)
    {
        $this->booksOnLoan[] = \Cassell\Library\Book\Book::rehydrate($bookWasBorrowed->getBookId(),$bookWasBorrowed->getBo
    }

    private function applyBookWasReturnedEvent(BookWasReturned $bookWasReturned)
    {
        // removes from array
    }

    public function borrow(Book $book, DateBorrowed $dateBorrowed)
    {
        if(count($this->booksOnLoan) >= self::BOOK_LIMIT) {
            throw new TooManyBooksOnLoanException();
        }

        $this->applyEvent(new BookWasBorrowed($dateBorrowed, $book->getBookId(), $this->getMemberId()));
    }
```

Libraries

- https://github.com/szjani/predaddy
- https://github.com/prooph
- https://github.com/qandidate-labs/broadway

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

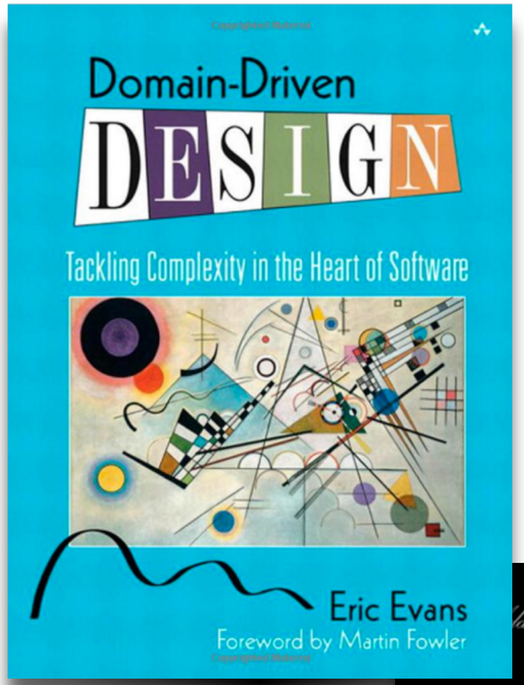
Working software over comprehensive documentation

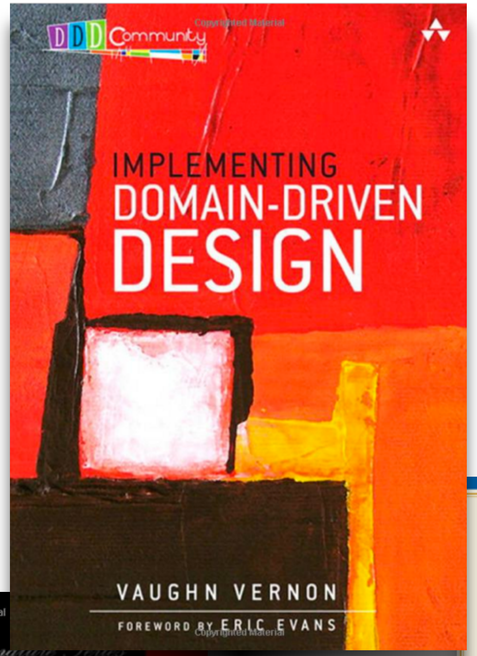
Customer collaboration over contract negotiation

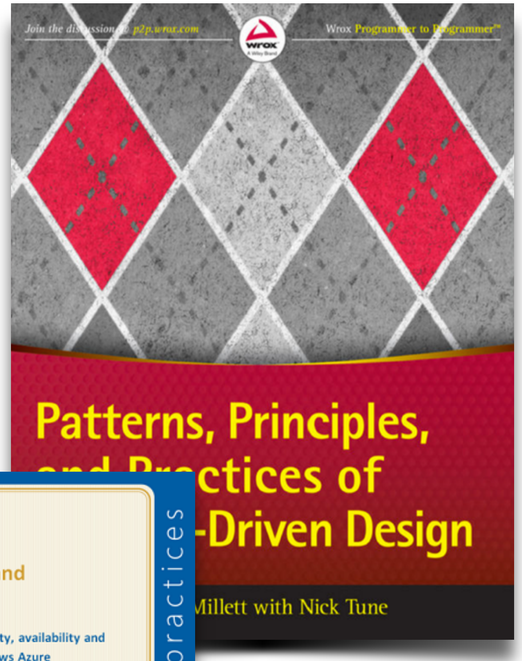
Responding to change over following a plan

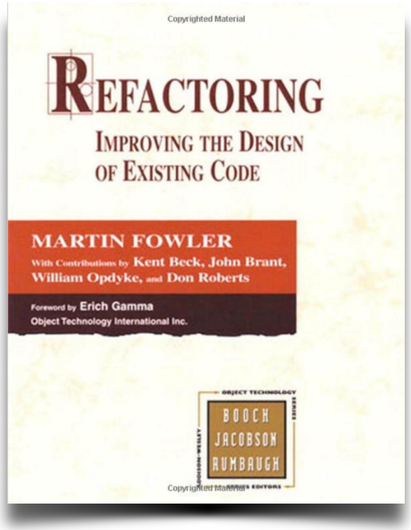
That is, while there is value in the items on the right, we value the items on the left more.

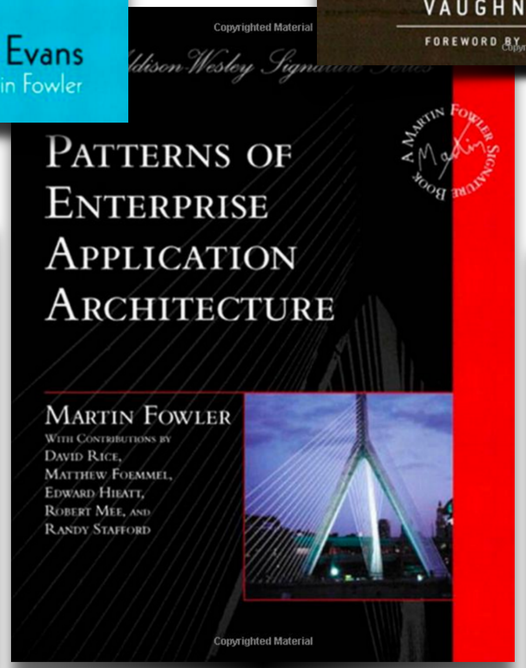
Learning More

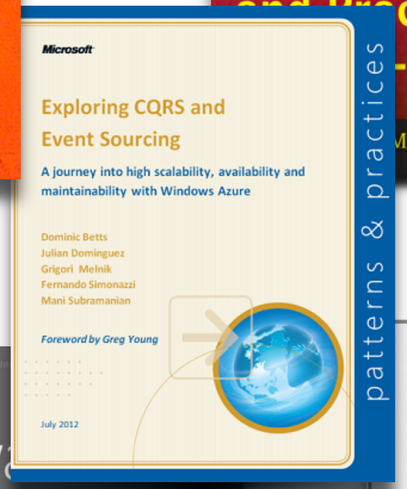

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

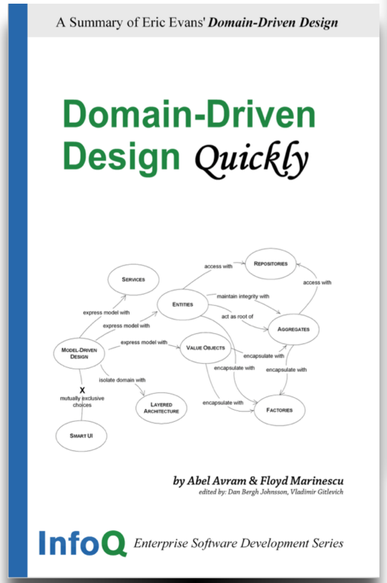

DDD Community
IMPLEMENTING
DOMAIN-DRIVEN
DESIGN
VAUGHN VERNON
FOREWORD BY ERIC EVANS

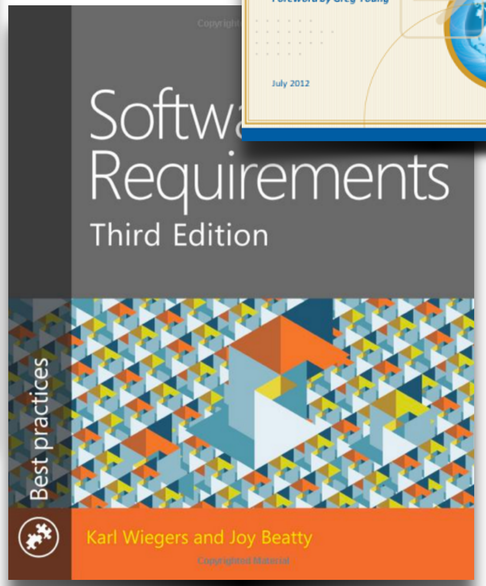

Wrox Programmer to Programmer
Patterns, Principles,
-Driven Design
Millett with Nick Tune

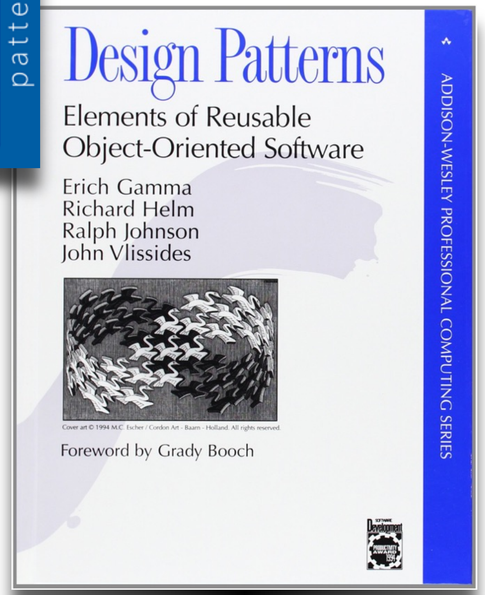

REFACTORING
IMPROVING THE DESIGN
OF EXISTING CODE
MARTIN FOWLER
With Contributions by Kent Beck, John Brant,
William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International Inc.
BOOCH
JACOBSON
RUMBAUGH

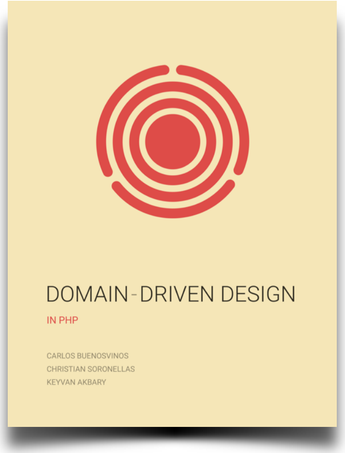

Addison-Wesley Signature Series
PATTERNS OF
ENTERPRISE
APPLICATION
ARCHITECTURE
MARTIN FOWLER
WITH CONTRIBUTIONS BY
DAVID RICE,
MATTHEW FOEMMEL,
EDWARD HIEATT,
ROBERT MEE, AND
RANDY STAFFORD
Copyrighted Material


Microsoft
Exploring CQRS and
Event Sourcing
A journey into high scalability, availability and
maintainability with Windows Azure
Dominic Betts
Julian Dominguez
Grigori Melnik
Fernando Simonazzi
Mani Subramanian
Foreword by Greg Young
July 2012
patterns & practices


A Summary of Eric Evans' Domain-Driven Design
Domain-Driven
Design Quickly
by Abel Avram & Floyd Marinescu
edited by: Dan Bergh Johnsson, Vladimir Gitlevich
InfoQ Enterprise Software Development Series


Software
Requirements
Third Edition
Best practices
Karl Wiegers and Joy Beatty


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


DOMAIN-DRIVEN DESIGN
IN PHP
CARLOS BUENOSVINOS
CHRISTIAN SORONELLAS
KEYVAN AKBARY

DevBookClub

https://www.youtube.com/user/devbookclub

Elephant in the Room

Podcast on PHP, Software Design, and Testing, by @everzet and @mathiasverraes

DOMAIN DRIVEN DESIGN EUROPE

JANUARY 26TH – 29TH 2016

BRUSSELS

IMPLEMENTING DOMAIN-DRIVEN DESIGN WORKSHOP

Learn how your team can use DDD to accelerate your software strategic initiatives for competitive advantage.

iDDD WORKSHOP

for{comprehension}

Go Reactive with Akka

Developing Microservices with Actors and DDD

2-Day Workshops
NYC: April 14-15
Provo: May 23-24
Copenhagen: June 20-21

3-Tage IDDD Workshop

(in deutscher Sprache)

Heidelberg Deutschland

13.-15. Juni 2016

3-Day IDDD Workshop Barcelona, ES 27-29 June 2016

for{comprehension}

NEW ONLINE TRAINING:

DISTILLING DOMAIN-DRIVEN DESIGN

by Vaughn Vernon

We offer both public and private workshops. If you prefer an on-site workshop, please contact us for details.

Mathias Verraes

Student of Systems
Meddler of Models
Labourer of Legacy

Upcoming

2016				
Topic	**Event**	**Type**	**Location**	**Date**
Introduction to Event Storming	CukeUp	Workshop	London, UK	April 14-15
CQRS/EventSourcing + EventStorming	NCrafts	Workshop	Paris, FR	May 11
TBD	NCrafts	Talk	Paris, FR	May 12-13
Experiencing Domain-Driven Design	Value Object Comm.V	Workshop	Berlin, DE	July 6-8
Tactical Domain-Driven Design	Laracon EU	Workshop	Amsterdam, NL	Aug
Experiencing Domain-Driven Design	Value Object Comm.V	Workshop	Amsterdam, NL	Sep 7-9
TBD	TYPO3 Developer Days	Talk	Nürnberg, DE	Sep

Older entries...

This repository Search Pull requests Issues Gist

PhpFriendsOfDdd / **state-of-the-union**

Watch 108 Star 436 Fork 47

Code Issues 4 Pull requests 0 Wiki Pulse Graphs

Describes various php Domain Driven Development initiatives all around the universe

59 commits **1** branch **0** releases **22** contributors

Branch: **master** New pull request Create new file Upload files Find file Clone or download

shouze Merge pull request #24 from jgimeno/master ... Latest commit `23c1e7d` on Dec 21, 2015

LICENSE	Beerware Licence	2 years ago
README.md	Update README.md	5 months ago

README.md

state-of-the-union

About

DDDinPHP Google Group

http://DDDinPHP.org

Topics Covered

- Ubiquitous Language
- Event Storming
- Modelling
- Value Objects
- Entities
- Aggregates
- Hexagonal Architecture
- CQRS
- Event Sourcing

https://joind.in/talk/99c3a