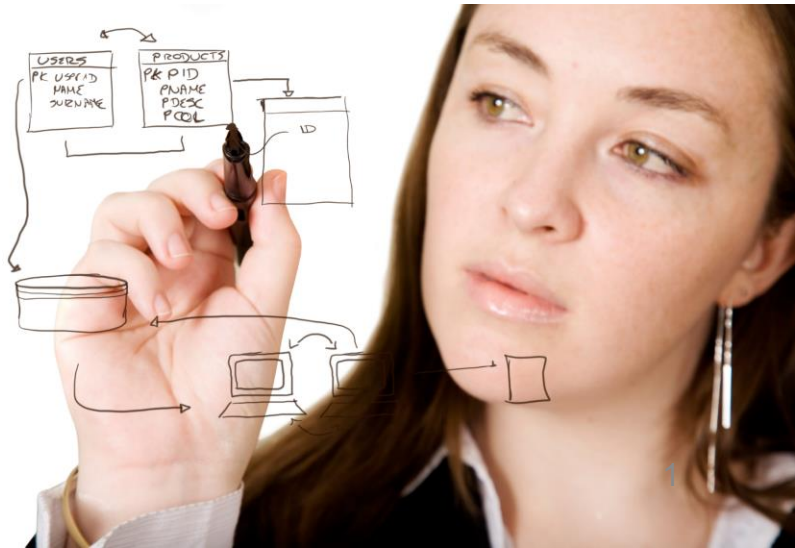
Azrieli College of Engineering Jerusalem
עזריאלי מכללה אקדמית להנדסה ירושלים

מודלים לפיתוח מערכות תוכנה
Software Systems Modeling

קורס 12003

סמסטר ב' תשע"ז

7. תיכון מונחה תחום וסיכום



ד"ר ראובן יגל
robi@post.jce.ac.il

הפעם

- DDD
- סיכום הקורס
- מצגות פרויקט ספר הקורס

Disseminating Architectural Knowledge on Open-Source Projects

A Case Study of the Book "Architecture of Open Source Applications"

Martin Robillard

McGill University

Nenad Medvidović

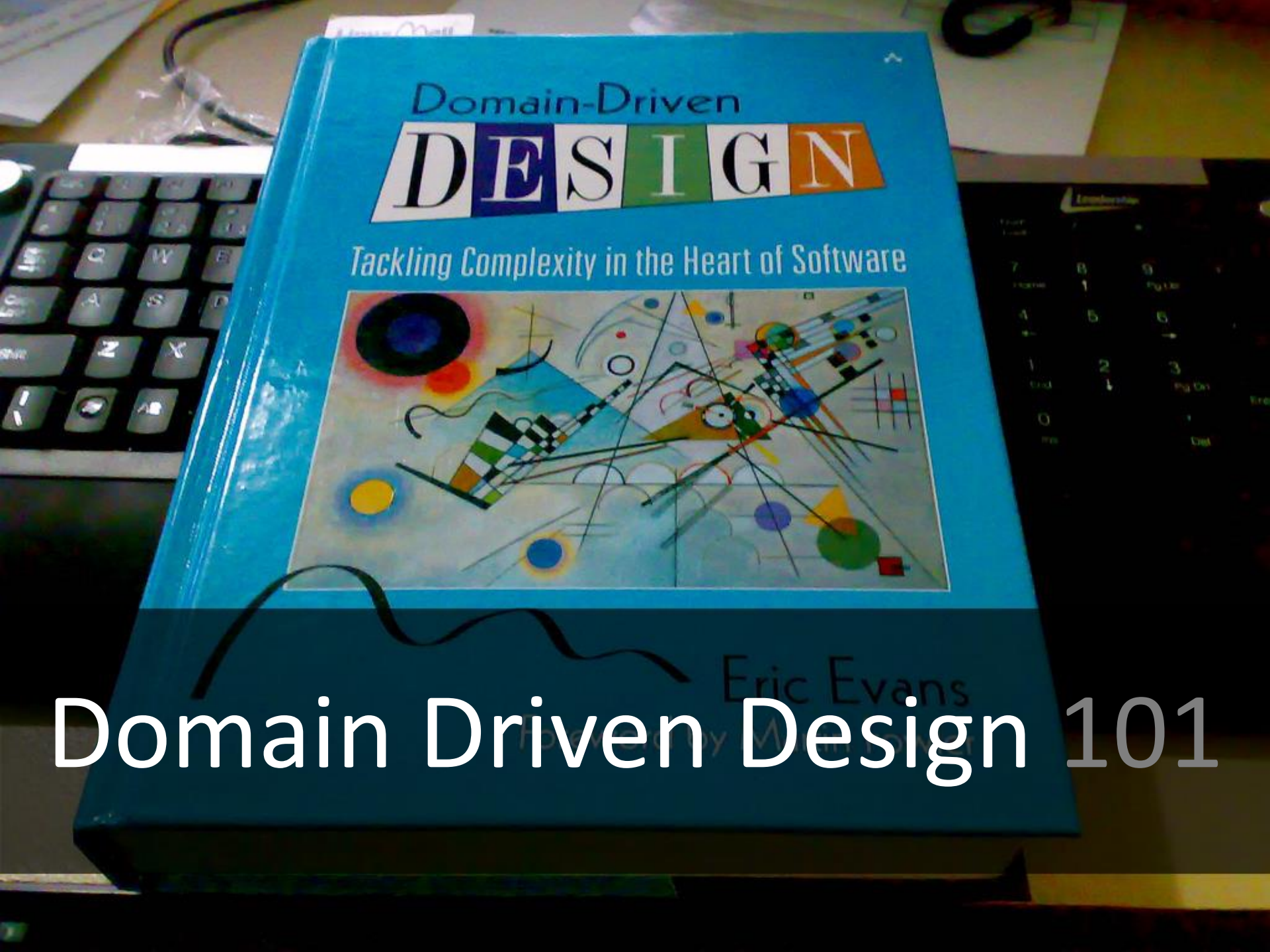
University of Southern California

Full-text of the paper at http://www.cs.mcgill.ca/~martin/papers/icse2016a.pdf

Presented at ICSE 2016 – 19 May 2016

Domain Driven Design

- Domain Driven Design (slideshare)

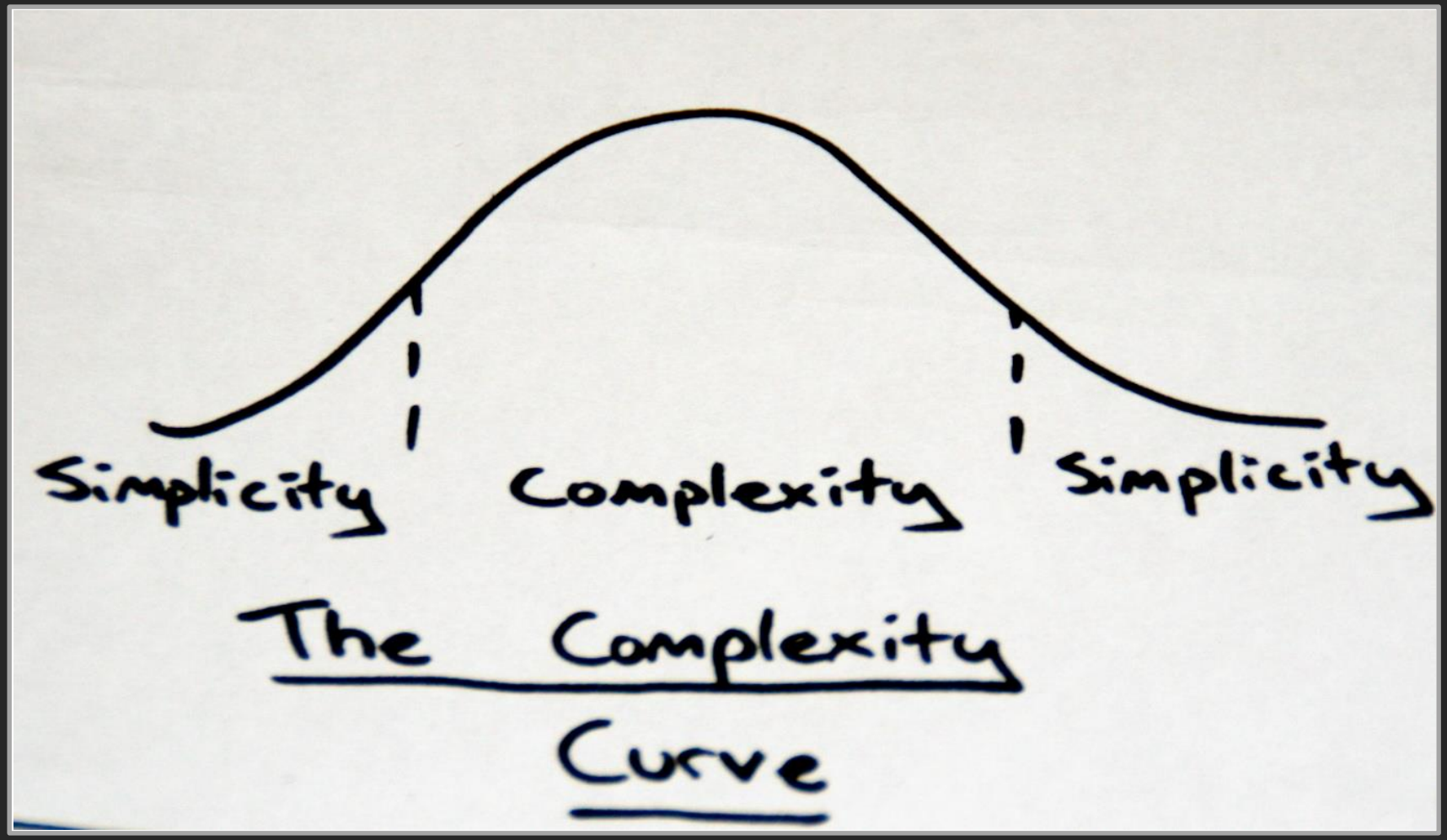


Domain Driven Design 101

Agenda

- Why
- Building blocks
  - Repositories, entities, specifications etc
- Putting it to practice
  - Dependency injection
  - Persistence
  - Validation
  - Architecture
- Challenges
- When not to use DDD
- Resources

Software is complicated

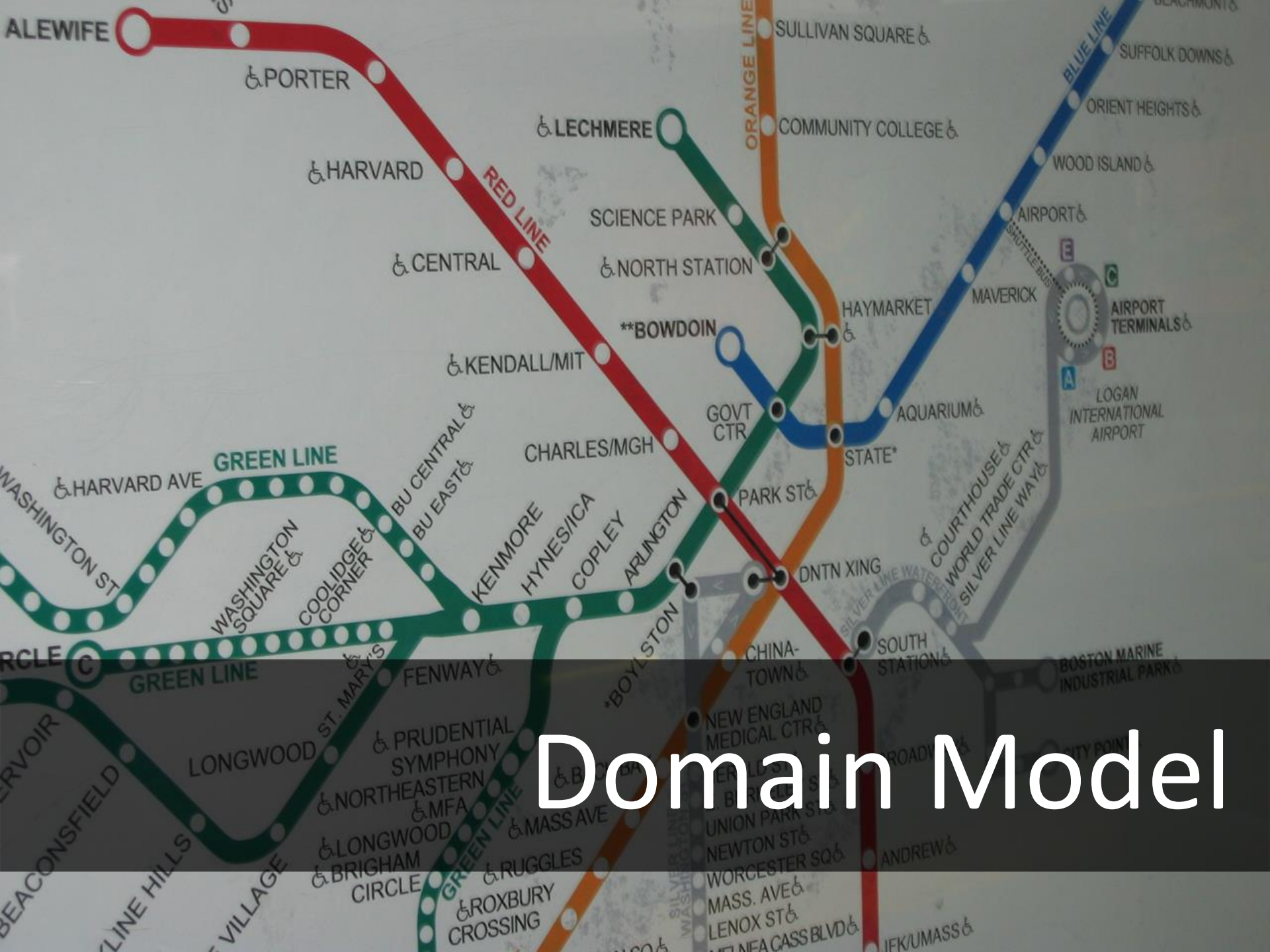


We solve complexity in software by distilling our problems

```
public bool CanBook(Cargo cargo, Voyage voyage)
{
    double maxBooking = voyage.Capacity * 1.1;
    if (voyage.BookedCargoSize + cargo.Size > maxBooking)
        return false;

    ...
}
```

```
public bool CanBook(Cargo cargo, Voyage voyage)
{
    if (!overbookingPolicy.IsAllowed(cargo, voyage))
        return false;

    ...
}
```

DDD is about making concepts explicit



Domain Model

Ubiquitous language

```
public interface ISapService
{
    double GetHourlyRate(int sapId);
}
```

A poor abstraction

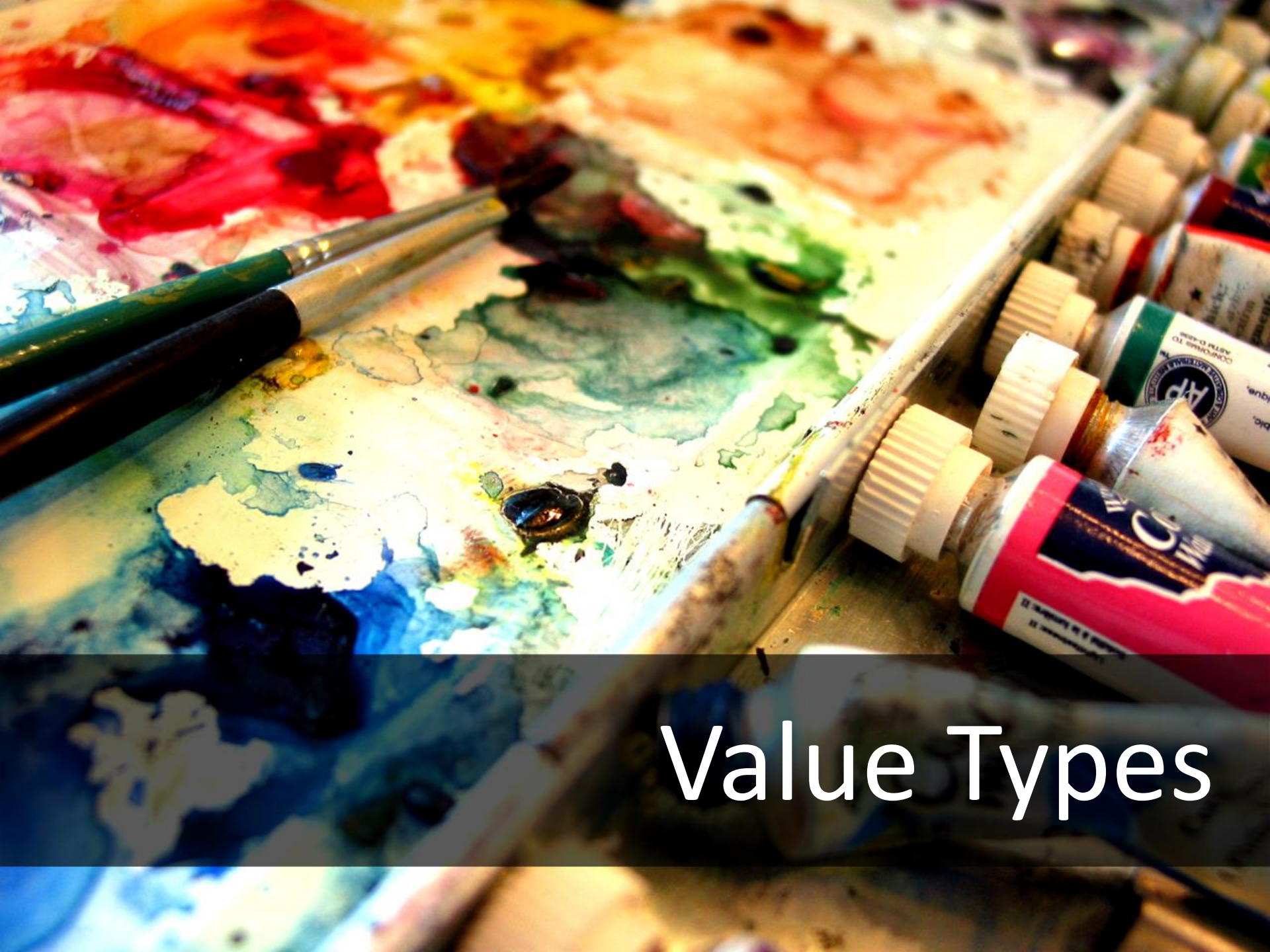
```
public interface IPayrollService
{
    double GetHourlyRate(Employee employee);
}
```

Intention-revealing interfaces

```
public class Employee
{
    void ApplyForLeave(DateTime start,
                       DateTime end,
                       ILeaveService leaves)
    {
        ...
    }
}
```


Domain Expert

Entities



Value Types

```
public class Employee : IEquatable<Employee>
{
    public bool Equals(Employee other)
    {
        return this.Id.Equals(other.Id);
    }
}
```

*Entities are the same if they have the same **identity***

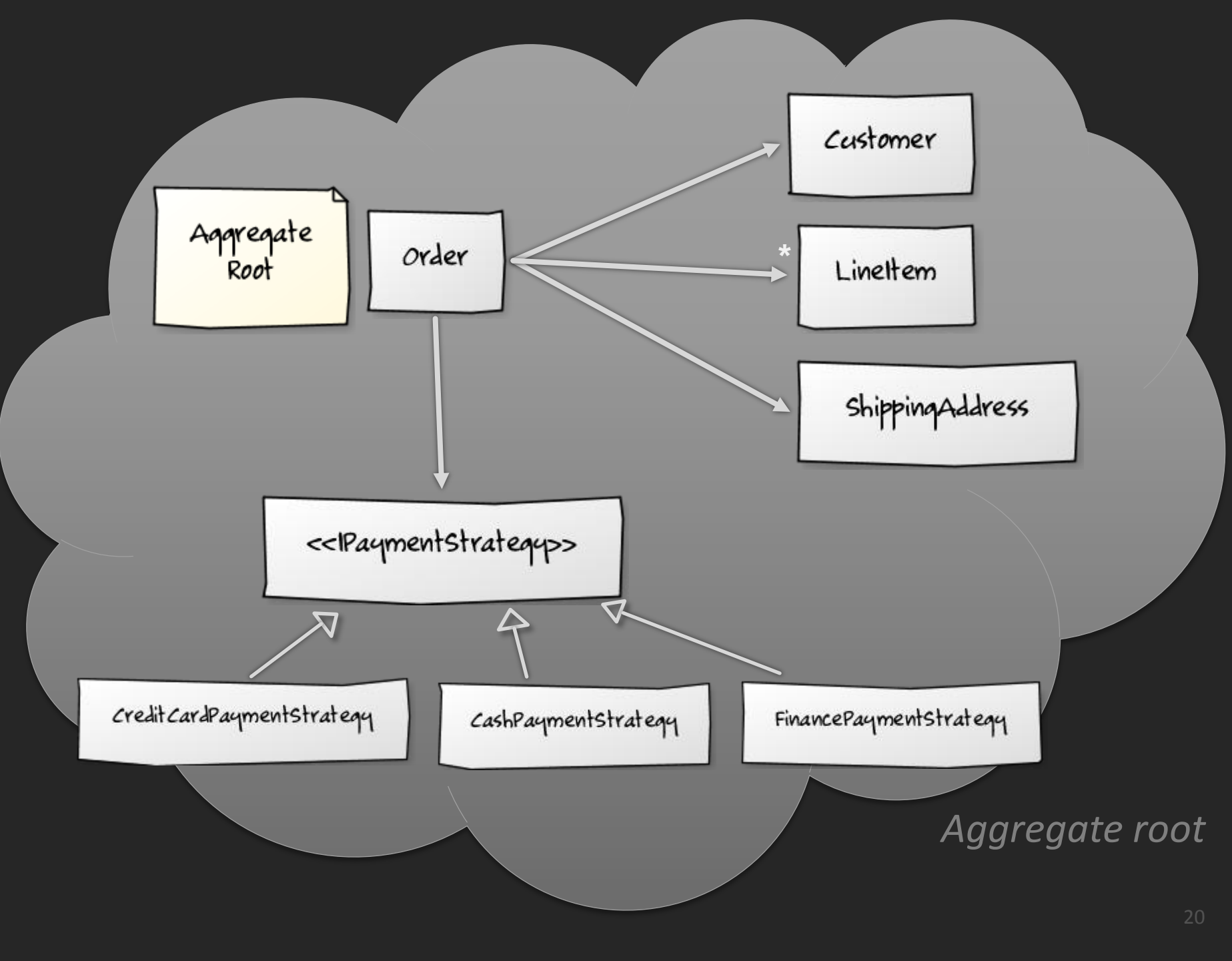
```
public class PostalAddress : IEquatable<PostalAddress>
{
    public bool Equals(PostalAddress other)
    {
        return this.Number.Equals(other.Number)
            && this.Street.Equals(other.Street)
            && this.PostCode.Equals(other.PostCode)
            && this.Country.Equals(other.Country);
    }
}
```

*Value Types are the same if they have the same **value***

```
public class Colour
{
    public int Red { get; private set; }
    public int Green { get; private set; }
    public int Blue { get; private set; }

    public Colour(int red, int green, int blue)
    {
        this.Red = red;
        this.Green = green;
        this.Blue = blue;
    }

    public Colour MixInTo(Colour other)
    {
        return new Colour(
            Math.Avg(this.Red, other.Red),
            Math.Avg(this.Green, other.Green),
            Math.Avg(this.Blue, other.Blue));
    }
}
```

*Value Types are **immutable***

Aggregates



Aggregate root

Repositories

```
public interface IEmployeeRepository
{
    Employee GetById(int id);
    void Add(Employee employee);
    void Remove(Employee employee);

    IEnumerable<Employee> GetStaffWorkingInRegion(Region region);
}
```

Repositories provide collection semantics and domain queries

Domain Services

```
public interface ITripService
{
    float GetDrivingDistanceBetween(Location a, Location b);
}
```

Specifications

```
class GoldCustomerSpecification : ISpecification<Customer>
{
    public bool IsSatisfiedBy(Customer candidate)
    {
        return candidate.TotalPurchases > 1000.0m;
    }
}
```

```
if (new GoldCustomerSpecification().IsSatisfiedBy(employee))
    // apply special discount
```

Specifications encapsulate a single rule

Specifications can be used…

to construct objects

```
var spec = new PizzaSpecification()
               .BasedOn(new MargaritaPizzaSpecification())
               .WithThickCrust()
               .WithSwirl(Sauces.Bbq)
               .WithExtraCheese();

var pizza = new PizzaFactory().CreatePizzaFrom(spec);
```

Constructing objects according to a specification

Specifications can be used…

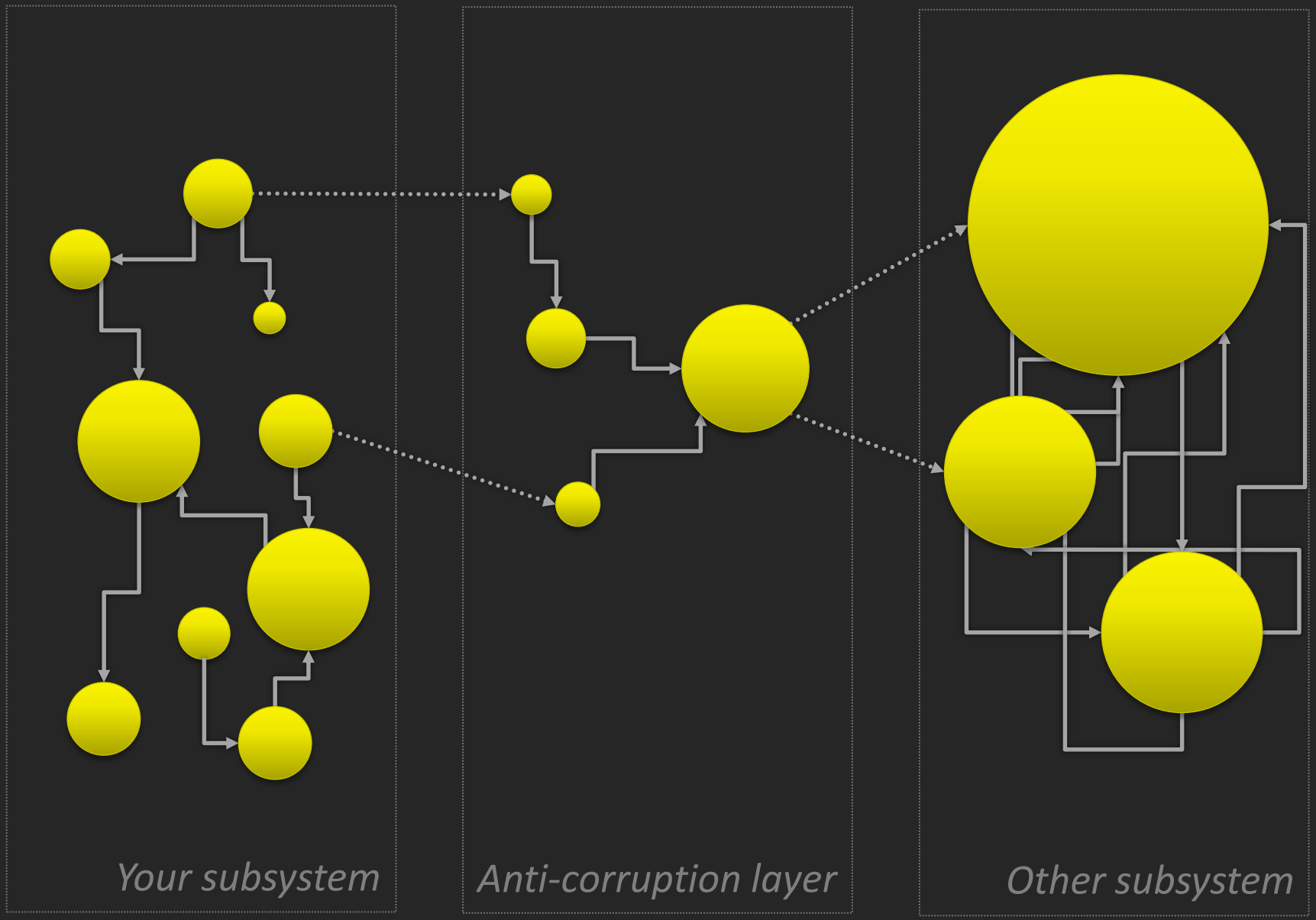
for querying

```
public interface ICustomerRepository
{
    IEnumerable<Customer> GetCustomersSatisfying(
        ISpecification<Customer> spec);
}
```

```
var goldCustomerSpec = new GoldCustomerSpecification();

var customers = this.customerRepository
    .GetCustomersSatisfying(goldCustomerSpec);
```

Querying for objects that match some specification

Anticorruption Layer


Your subsystem
Anti-corruption layer
Other subsystem

“Any 3rd party system that I have to integrate with was written by a drunken monkey typing with his feet.”

Oren Eini aka Ayende

Bounded Context

```
public class Lead
{
    public IEnumerable<Opportunity> Opportunities { get; }
    public Person Contact { get; }
}
```

```
public class Client
{
    public IEnumerable<Invoice> GetOutstandingInvoices();
    public Address BillingAddress { get; }
    public IEnumerable<Order> PurchaseHistory { get; }
}
```

```
public class Customer
{
    public IEnumerable<Ticket> Tickets { get; }
}
```

Dependency Injection

```
public interface INotificationService
{
    void Notify(Employee employee, string message);
}
```

An interface defines the model

```
public class EmailNotificationService : INotificationService
{
    void Notify(Employee employee, string message)
    {
        var message = new MailMessage(employee.Email, message);
        this.smtpClient.Send(message);
    }
}
```

Far away, a concrete class satisfies it

```
public class LeaveService
{
    private readonly INotificationService notifications;

    public LeaveService(INotificationService notifications)
    {
        this.notifications = notifications;
    }

    public void TakeLeave(Employee employee, DateTime start,
                          DateTime end)
    {
        // do stuff

        this.notifications.Notify(employee, "Leave approved.");
    }
}
```

Dependencies are injected at runtime

Persistence Ignorance

“…ordinary classes where you focus on the business problem at hand without adding stuff for infrastructure-related reasons… nothing else should be in the Domain Model.”

```
public class Customer
{
    public int Id { get; private set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }

    public IEnumerable<Address> Addresses { get; }
    public IEnumerable<Order> Orders { get; }

    public Order CreateOrder(ShoppingCart cart)
    {
        ...
    }
}
```

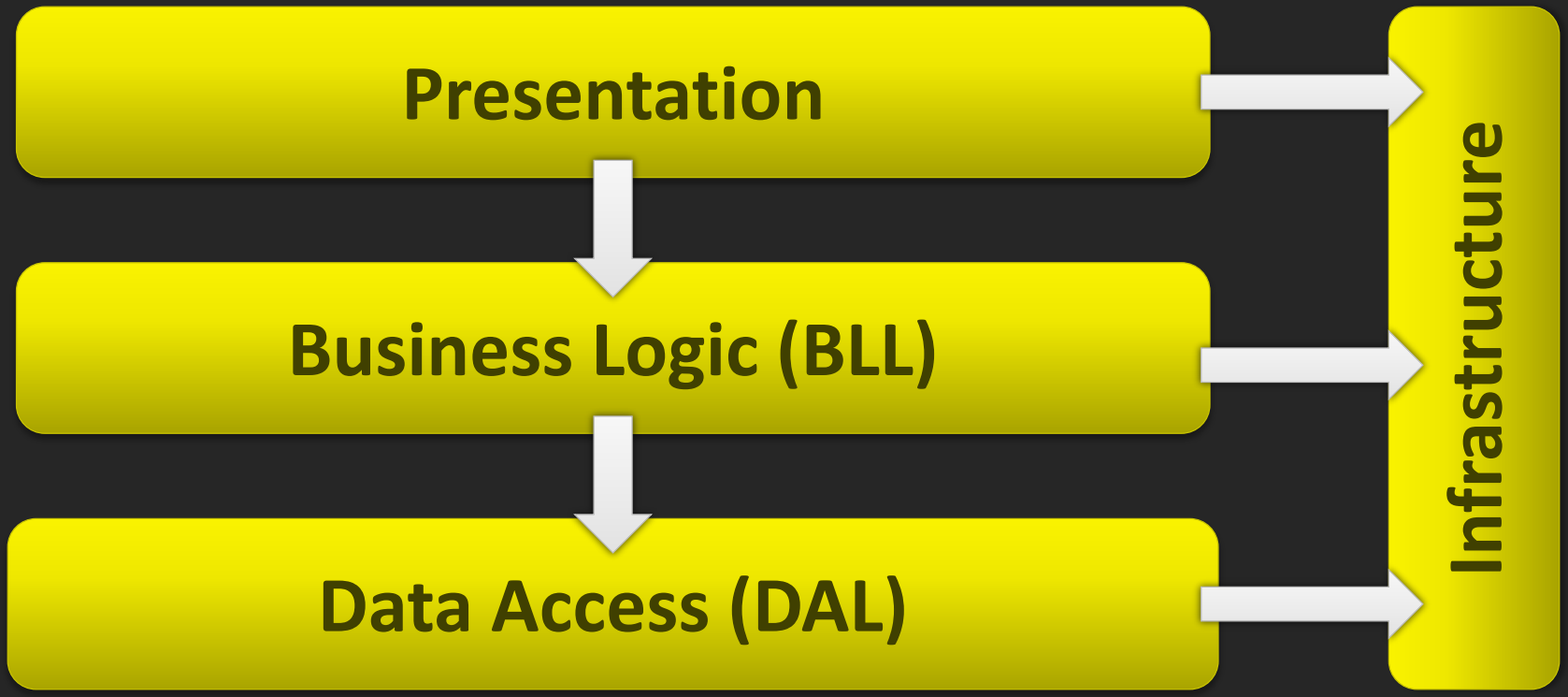
Plain Old CLR Object (POCO)

```
[global::System.Data.Objects.DataClasses.EdmEntityTypeAttribute(NamespaceName="AdventureWorksLTModel",
Name="Customer")]
    [global::System.Runtime.Serialization.DataContractAttribute(IsReference=true)]
    [global::System.Serializable()]
    public partial class Customer : global::System.Data.Objects.DataClasses.EntityObject
    {
        [global::System.Data.Objects.DataClasses.EdmScalarPropertyAttribute(EntityKeyProperty=true,
IsNullable=false)]
        [global::System.Runtime.Serialization.DataMemberAttribute()]
        public int CustomerID
        {
            get
            {
                return this._CustomerID;
            }
            set
            {
                this.OnCustomerIDChanging(value);
                this.ReportPropertyChanging("CustomerID");
                this._CustomerID =
global::System.Data.Objects.DataClasses.StructuralObject.SetValidValue(value);
                this.ReportPropertyChanged("CustomerID");
                this.OnCustomerIDChanged();
            }
        }
        private int _CustomerID;
        partial void OnCustomerIDChanging(int value);
        partial void OnCustomerIDChanged();
```

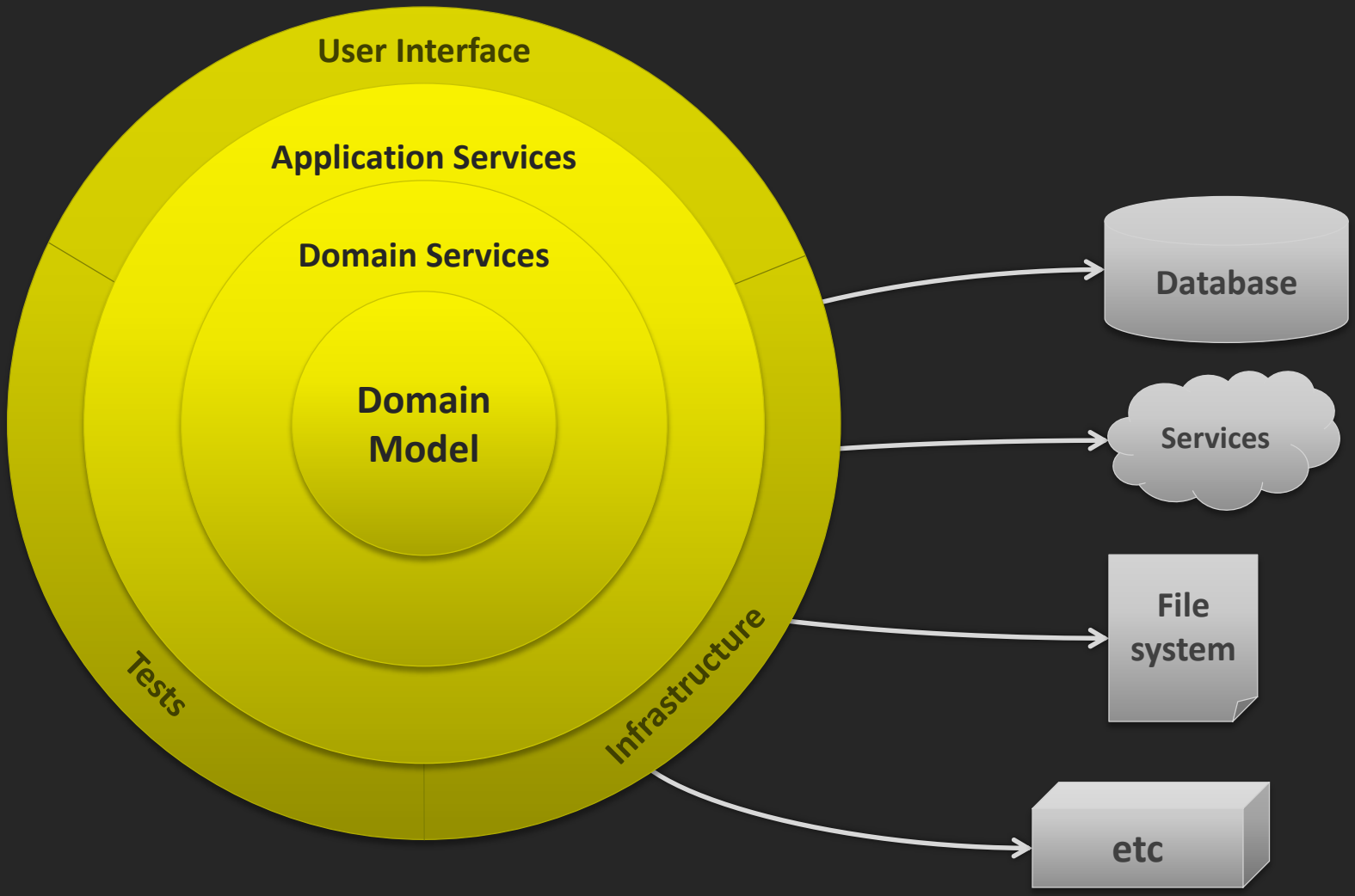
This is not a POCO.

Architecture

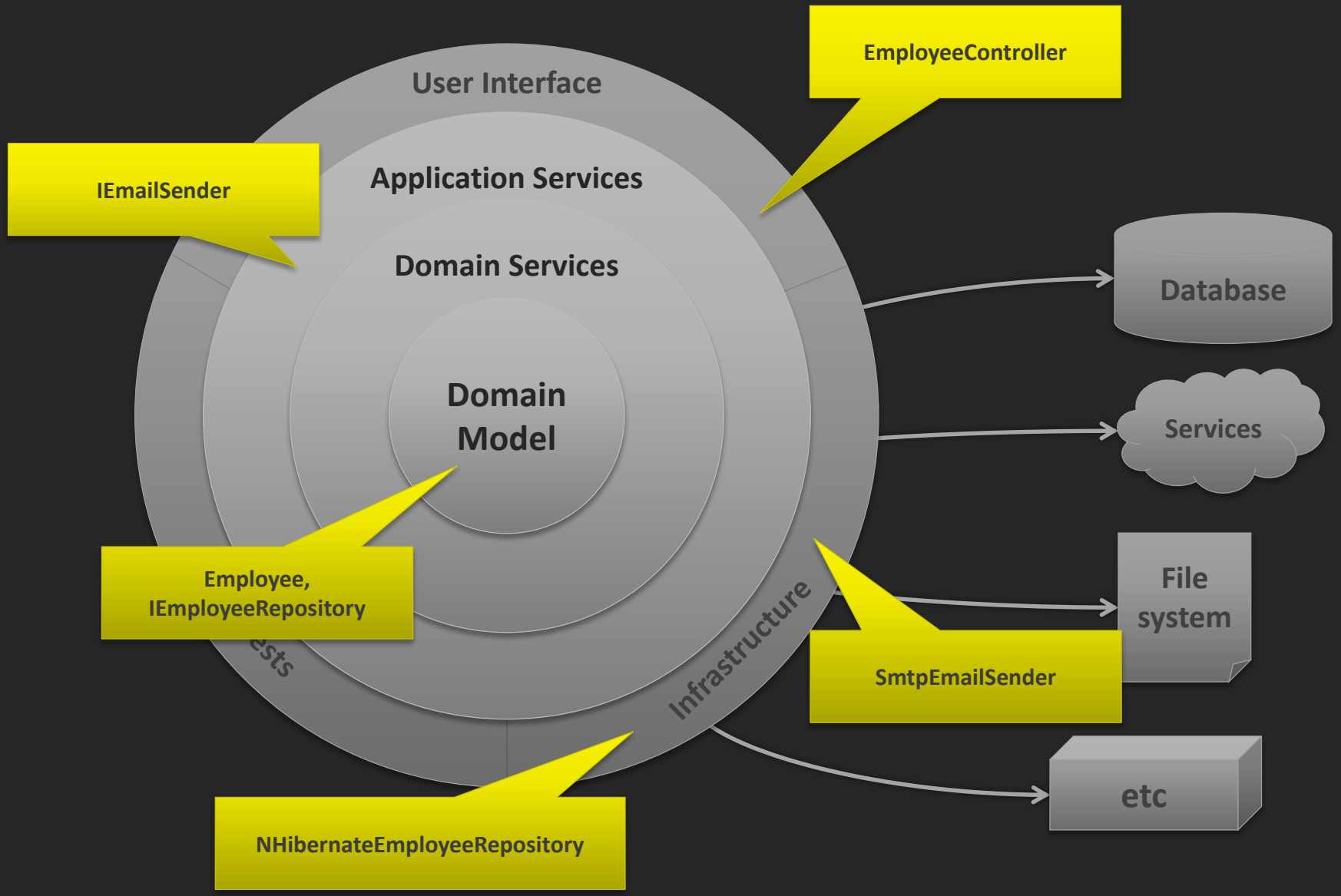
Traditional Architecture



Onion Architecture



Onion Architecture



Validation

Validation Examples

Input validation

Business domain

- Is the first name filled in?
- Is the e-mail address format valid?
- Is the first name less than 255 characters long?
- Is the chosen username available?
- Is the password strong enough?
- Is the requested book available, or already out on loan?
- Is the customer eligible for this policy?

```
public class PersonRepository : IPersonRepository
{
    public void Save(Person customer)
    {
        if (!customer.IsValid())
            throw new Exception(...)
    }
}
```

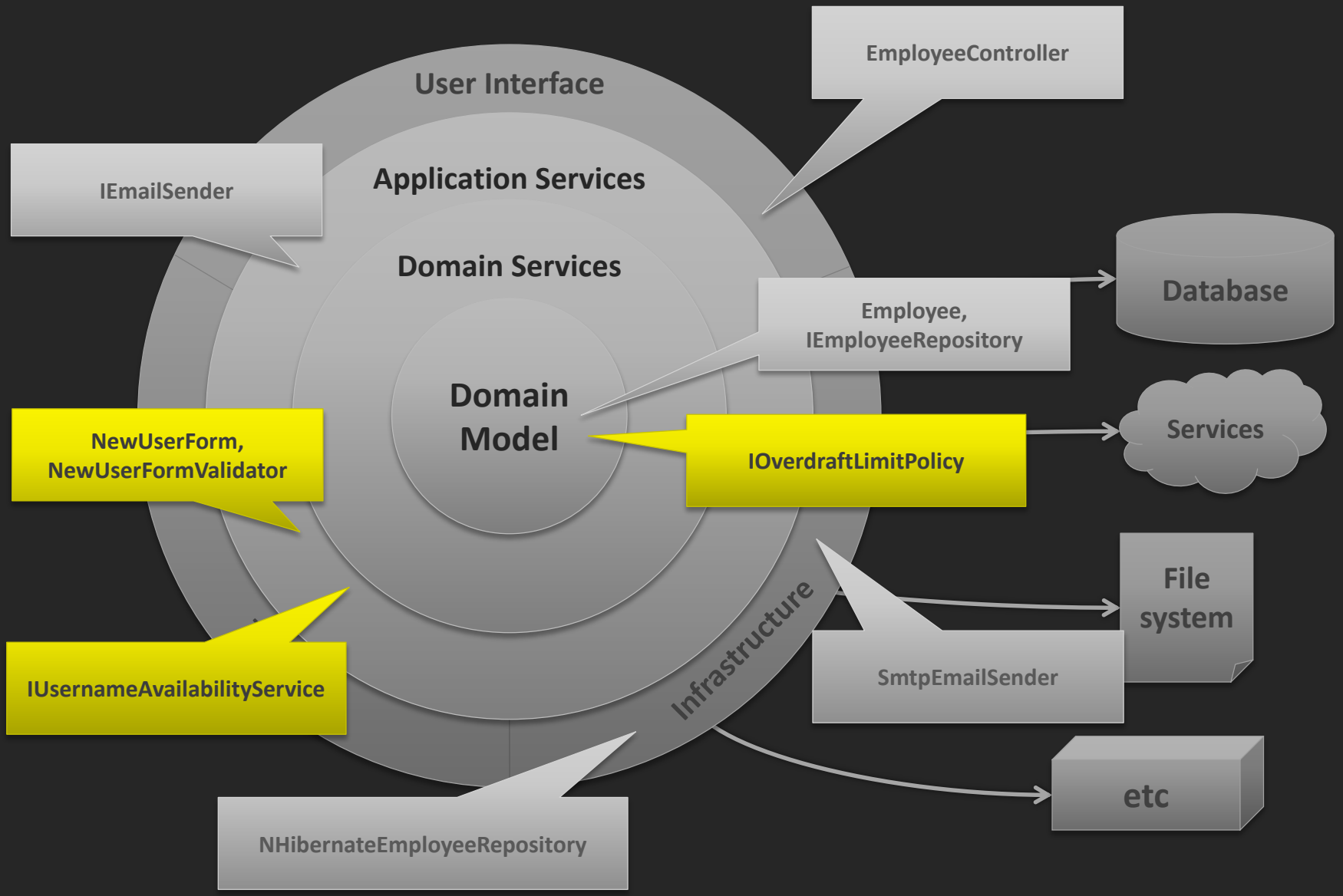
validation and persistence anti-patterns


The golden rule for validation:
The Domain Model is always
in a valid state

```
public class NewUserFormValidator : AbstractValidator<NewUserForm>
{
    IUsernameAvailabilityService usernameAvailabilityService;

    public NewUserFormValidator()
    {
        RuleFor(f => f.Email).EmailAddress();

        RuleFor(f => f.Username).NotEmpty().Length(1, 32)
            .WithMessage("Username must be between 1 and 32 characters");

        RuleFor(f => f.Url).Must(s => Uri.IsWellFormedUriString(s))
            .Unless(f => String.IsNullOrEmpty(f.Url))
            .WithMessage("This doesn't look like a valid URL");

        RuleFor(f => f.Username)
            .Must(s => this.usernameAvailabilityService.IsAvailable(s))
            .WithMessage("Username is already taken");
    }
}
```

separation of validation concerns with FluentValidation

Where validation fits

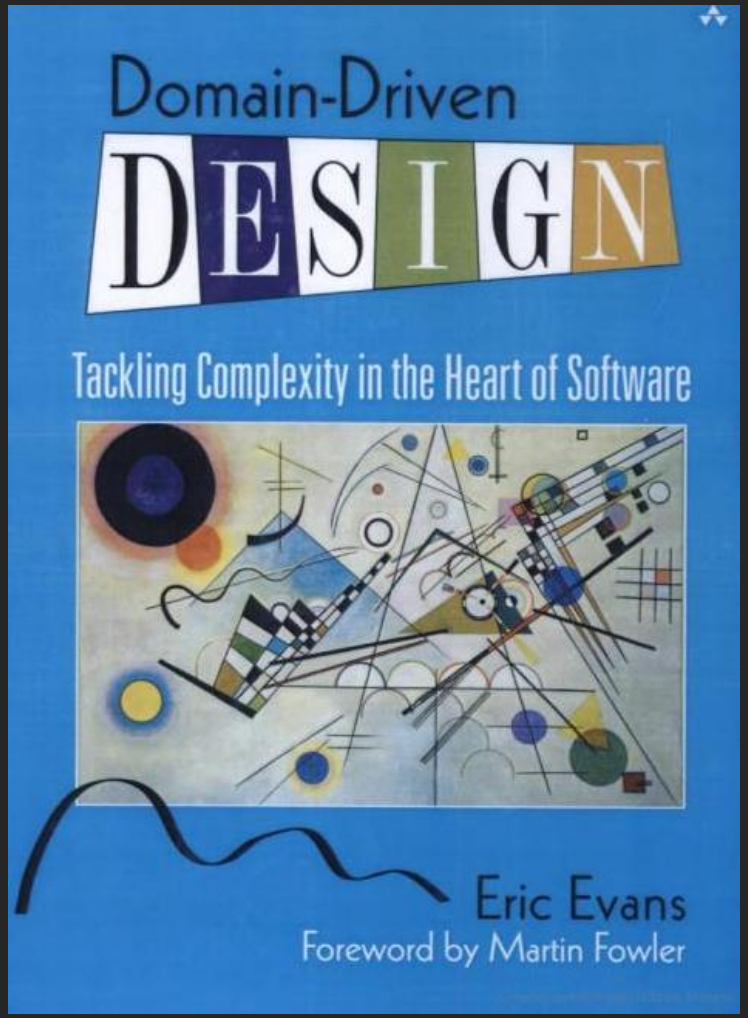


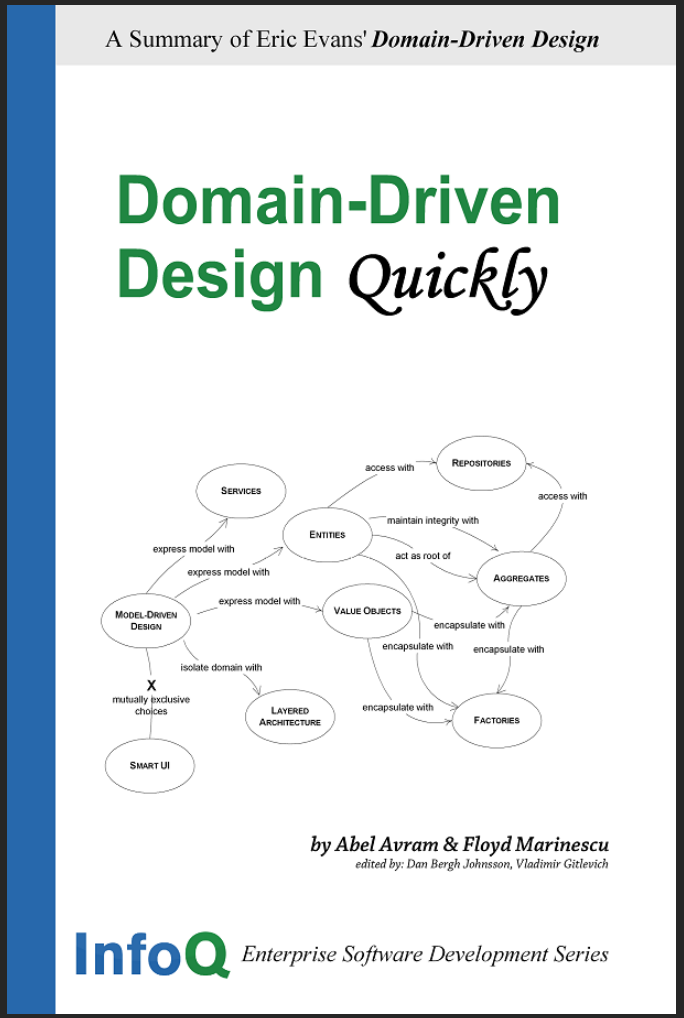

Challenges

When DDD isn’t appropriate


Benefits

Books


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


A Summary of Eric Evans' Domain-Driven Design
Domain-Driven Design Quickly
Services
Repositories
Entities
Aggregates
Model-Driven Design
Value Objects
Layered Architecture
Factories
Smart UI
access with
maintain integrity with
act as root of
express model with
isolate domain with
encapsulate with
mutually exclusive choices
by Abel Avram & Floyd Marinescu
edited by: Dan Bergh Johnsson, Vladimir Gitlevich
InfoQ Enterprise Software Development Series

Links

Domain Driven Design mailing list

- http://tech.groups.yahoo.com/group/domaindrivendesign/

ALT.NET mailing list

- http://tech.groups.yahoo.com/group/altdotnet/

DDD Step By Step

- http://dddstepbystep.com/

Domain Driven Design Quickly (e-book)

- http://www.infoq.com/minibooks/domain-driven-design-quickly

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

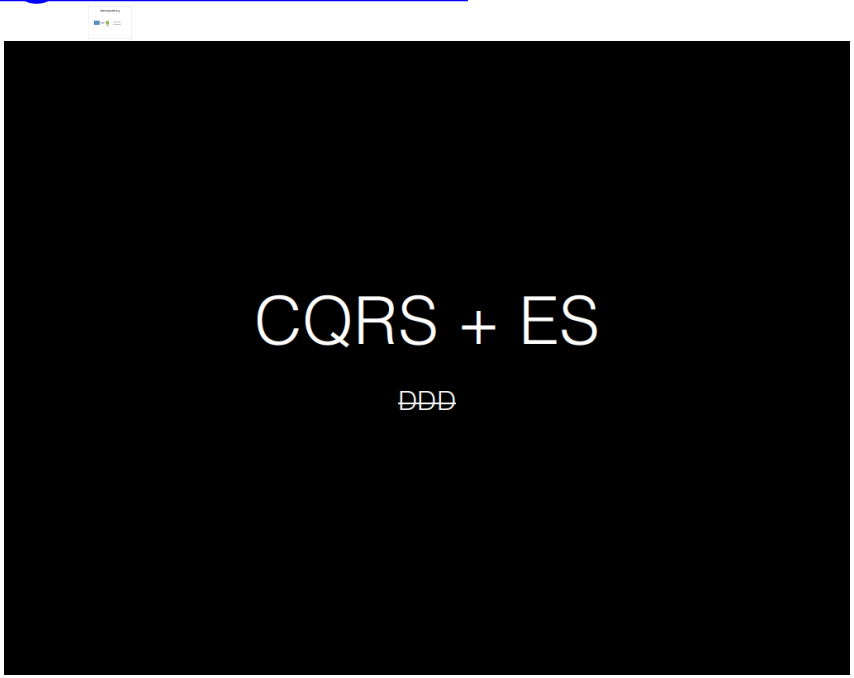
Martin Fowler

דוגמאות קוד

- https://github.com/andreazevedo/petstore-ddd-csharp
- https://github.com/ardalis/ddd-vet-sample (a starter?)
- https://github.com/VaughnVernon/IDDD_Samples_NET
- http://dddsamplenet.codeplex.com/

CQRS / EventSourcing

- http://www.slideshare.net/mbild/cqrs-event-sourcing-28292586

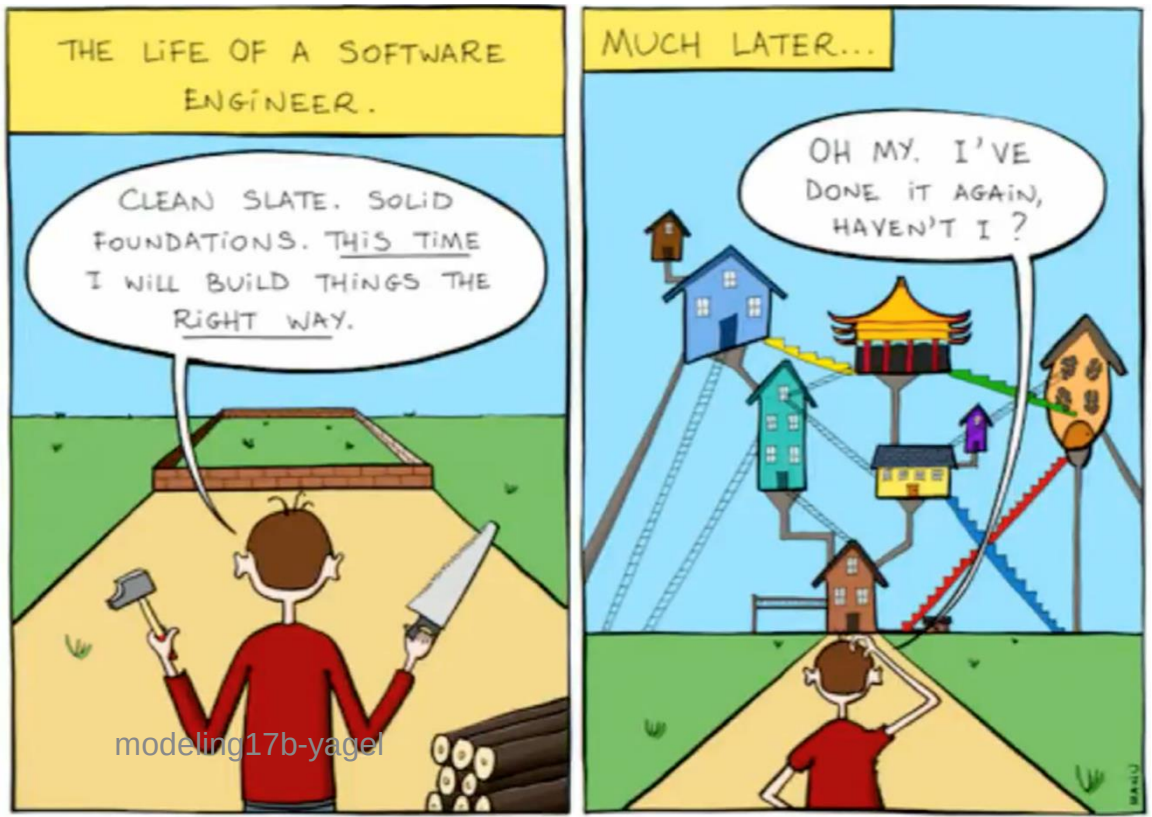


DDD / Event Sourcing Katas

- https://github.com/DevLyon/mixter
- https://github.com/dgadd/TDD-Kata-for-DDD
- http://www.jspcore.com/mars-rover-kata/
- http://codingsolutions.blogspot.co.il/2011/10/short-ddd-kata.html

סיכום הקורס

- נושאים שעברנו – מאגר הקורס – https://github.com/jce-il/sw-modeling-2017b
- מטלות
- סיכום ודיון
- מה הלאה



מצגות ASOSMA

- https://github.com/jce-il/ASOSMA/pulse

תודה רבה