SBA Research

Type-Driven Domain Modeling

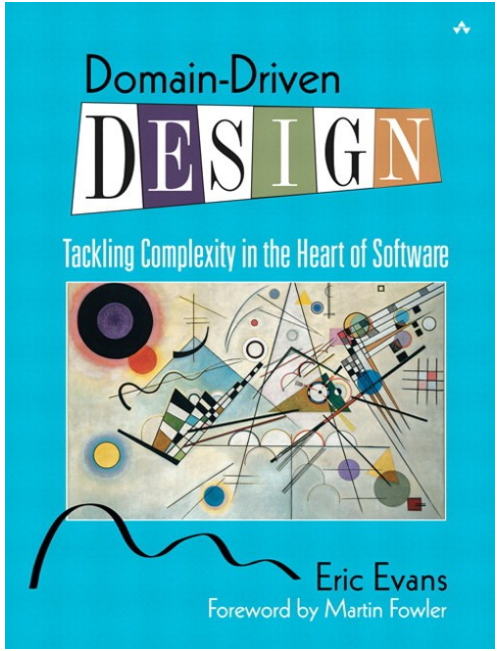
Use the types, Luke!

Security Meetup – 2022-01-26

Bundesministerium Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie
Bundesministerium Digitalisierung und Wirtschaftsstandort
FFG Forschung wirkt.
wirtschafts agentur wien Ein Fonds der Stadt Wien
European Commission
FWF Der Wissenschaftsfonds.
netidee OPEN INNOVATIONS

Introduction

Wisdoms from the Blue Book

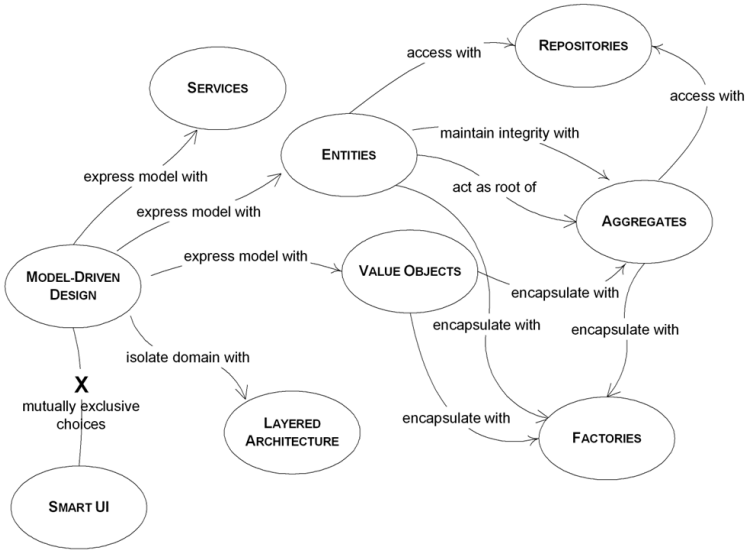


Domain-Driven Design (DDD)

- Code and structure should match the business domain
- Primary focus on core domain and business logic
- Iteratively refine concepts by consulting domain experts
- Uses ubiquitous language that everyone in the domain understands

Popular Concepts

- Entity
- Value object
- Aggregate
- Bounded Contexts
- Repository
- Command Query Responsibility Segregation



Source: Domain-Driven Design – Eric Evans

Type-Driven Domain Modeling

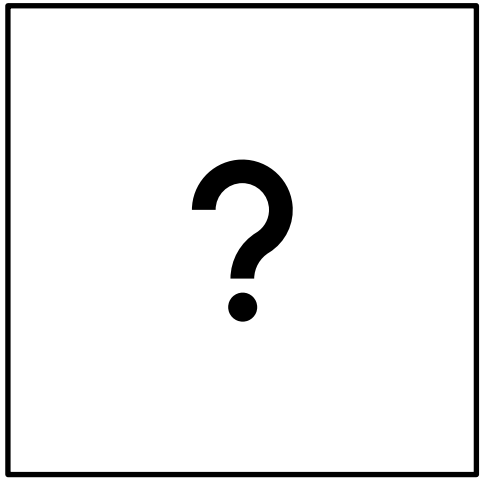
- Encode business rules into types
- Make illegal state unrepresentable
- Reason without looking at implementations
- Prevent security vulnerabilities
- Offload work to the compiler

Language Support

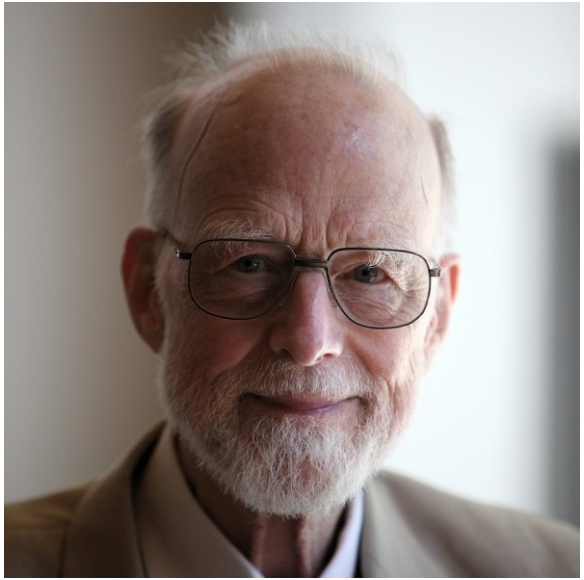
- Any language can be used for DDD
- But some provide more powerful type systems
- Nice to have: Sum Types, Pattern Matching

- Examples:
  - Haskell
  - Elm
  - OCaml
  - F#
  - Rust
  - Scala
  - Reason
  - TypeScript
  - C# (7)
  - Java (17)

Null



Null



Source: https://commons.wikimedia.org/wiki/File:Sir_Tony_Hoare_IMG_5125.jpg

"I call it my billion-dollar mistake. It was the invention of the null reference in 1965."
Tony Hoare

Make the Absence of Values Explicit (Haskell)

```haskell
data Maybe a = Nothing | Just a

lookup :: String -> [(String, String)] -> Maybe String
lookup key list = case list of
    []             -> Nothing
    ((k, v):rest) -> if key == k then Just v else lookup key rest

phonebook :: [(String, String)]
phonebook = [("Alice", "01889 985333"), ("Bob", "01788 665242")]

getNumber :: String -> [(String, String)] -> String
getNumber name list = case lookup name list of
    Nothing    -> "Could not find a number for " <> name
    Just value -> name <> "’s number is: " <> value
```

Modeling a Contact Type

Because not everything is a string

Demo Time

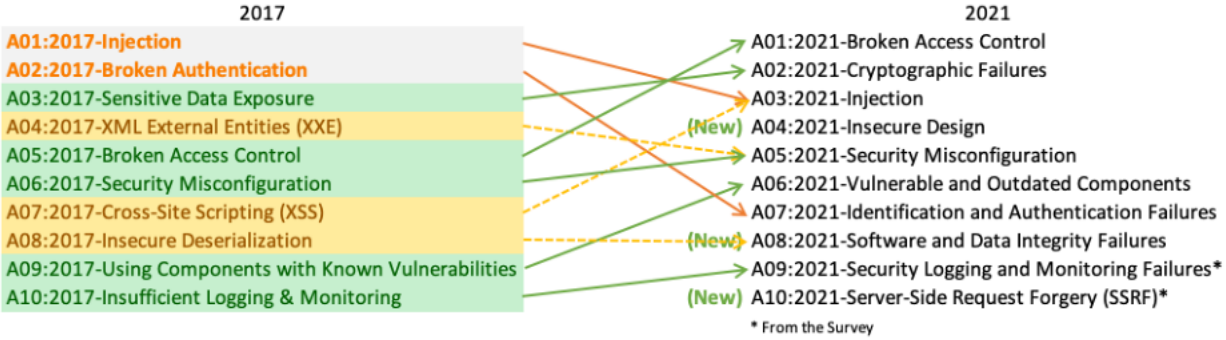
Domain modeling in F#

Typed Security

Preventing vulnerabilities by design

OWASP Top Ten (2021)

- A01:2021-Broken Access Control
- A03:2021-Injection
- A04:2021-Insecure Design



Source: https://owasp.org/Top10/

OWASP API Security Top 10 (2019)

- API1:2019 Broken Object Level Authorization
- API3:2019 Excessive Data Exposure
- API5:2019 Broken Function Level Authorization
- API6:2019 Mass Assignment
- API8:2019 Injection

Demo Time

Haskell

Fighting broken access control attacks

Fighting Injection Attacks (Haskell)

```haskell
import qualified Database.SQLite.Simple as SQL

main = SQL.withConnection "products.db" $ \conn -> do
    putStrLn "Search by product name:"
    pname    <- getLine
    products <- getProductsByName conn pname
    putStrLn ("Here is the data: " ++ show products)

-- SQL.query :: SQL.Connection -> SQL.Query -> args -> IO [result]

getProductsByName :: SQL.Connection -> String -> IO [Product]
getProductsByName conn pname =
    SQL.query conn (SQL.Query "SELECT * FROM products WHERE product_name=?") (pname)
```

Fighting Excessive Data Exposure (Java)

```
1.  public class User {
2.      private final Id<User> id;
3.      private Name50 firstName, lastName;
4.      private Birthdate dateOfBirth:
5.      private PasswordHash passwordHash;
6.      // Constructor, getters, setters, domain logic, etc.
7.  }
8.  public class UserViewModel {
9.      private final String fullName;
10.     private final DateTime dateOfBirth;
11.     // Constructor, getters, mapping from entity to dto and vice versa
12. }
13. public List<UserViewModel> getUsers(...) {
14.     List<User> users = userService.getUsers(...);
15.     return users.stream().map(UserViewModel::entityToViewModel).collect(Collectors.toList());
16. }
```

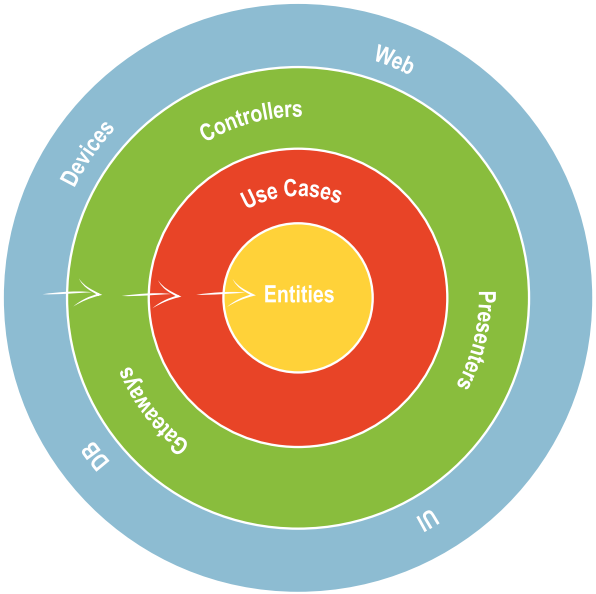
Fighting XSS Attacks (Elm)

```
1.  -- div : List (Attribute msg) -> List (Html msg) -> Html msg
2.  -- text : String -> Html msg

3.  userComponent : String -> Int -> Html msg
4.  userComponent userName userAge =
5.      div [ class "user-component" ]
6.          [ text userName
7.          , text (String.fromInt userAge)
8.          ]

9.  -- <div class="user-component">Foo42</div>
```

Choosing the Right™ Architecture

Building on SOLID principles

Hexagonal / Onion / Clean Architecture



Source: https://www.4soft.de/blog/2019/clean-architecture/

Example Repository: PackIT (C#)

- PackIT.Api
  - Controllers
  - Properties
  - appsettings.Development.json
  - appsettings.json
  - PackIT.Api.csproj
  - Program.cs
  - Startup.cs
- PackIT.Application
  - Commands
  - DTO
  - Exceptions
  - Queries
  - Services
  - Extensions.cs
  - PackIT.Application.csproj
- PackIT.Domain
  - Consts
  - Entities
  - Events
  - Exceptions
  - Factories
  - Policies
  - Repositories
  - ValueObjects
  - PackIT.Domain.csproj
- PackIT.Infrastructure
  - EF
  - Logging
  - Services
  - Extensions.cs
  - PackIT.Infrastructure.csproj
- PackIT.Shared
- PackIT.Shared.Abstractions

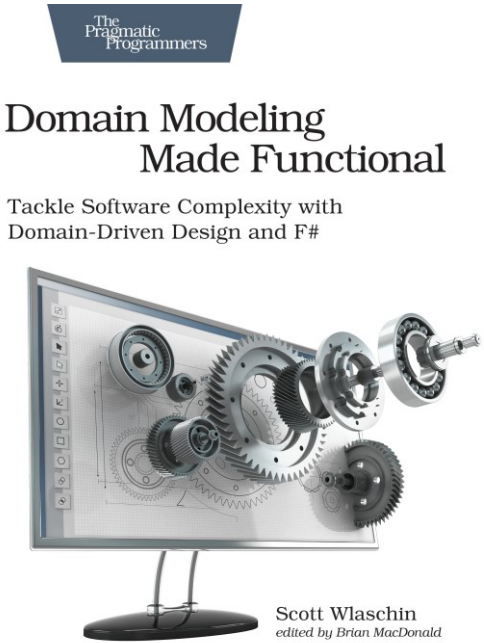
- PackIT.Application
  - Commands
    - Handlers
    - AddPackingItem.cs
    - CreatePackingListWithItems.cs
    - PackItem.cs
    - RemovePackingItem.cs
    - RemovePackingList.cs

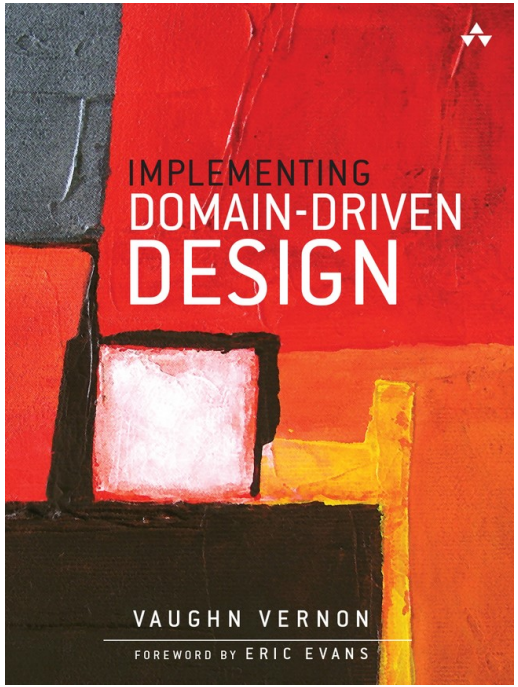
```csharp
using PackIT.Domain.Exceptions;

namespace PackIT.Domain.ValueObjects
{
    public record PackingItem
    {
        public string Name { get; }
        public uint Quantity { get; }
        public bool IsPacked { get; init; }

        public PackingItem(string name, uint quantity, bool isPacked = false)
        {
            if (string.IsNullOrWhiteSpace(name))
            {
                throw new EmptyPackingListItemNameException();
            }

            Name = name;
            Quantity = quantity;
            IsPacked = isPacked;
        }
    }
}
```

```csharp
using System.Threading.Tasks;
using PackIT.Application.Exceptions;
using PackIT.Domain.Repositories;
using PackIT.Domain.ValueObjects;
using PackIT.Shared.Abstractions.Commands;

namespace PackIT.Application.Commands.Handlers
{
    internal sealed class AddPackingItemHandler : ICommandHandler<AddPackingItem>
    {
        private readonly IPackinglistRepository _repository;

        public AddPackingItemHandler(IPackingListRepository repository)
            => _repository = repository;

        public async Task HandleAsync(AddPackingItem command)
        {
            var packingList = await _repository.GetAsync(command.PackingListId);

            if (packingList is null)
            {
                throw new PackingListNotFoundException(command.PackingListId);
            }

            var packingItem = new PackingItem(command.Name, command.Quantity);
            packingList.AddItem(packingItem);

            await _repository.UpdateAsync(packingList);
        }
    }
}
```


Source: https://www.youtube.com/watch?v=NzcZcim9tp8

Book Recommendations





Key Takeaways

- Make illegal state unrepresentable
- Encode business rules in your types
- Parse, don’t validate
- Use the compiler to your advantage
- Write readable code for domain experts
- Eliminate security vulnerabilities by design

Michael Koppmann

SBA Research

Floragasse 7, 1040 Wien

mkoppmann@sba-research.org

Bundesministerium Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie

Bundesministerium Digitalisierung und Wirtschaftsstandort

FFG Forschung wirkt.

wirtschafts agentur wien Ein Fonds der Stadt Wien

European Commission

FWF Der Wissenschaftsfonds.

netidee OPEN INNOVATIONS